Problem Set 7
CS 373: Theory of Computation

Assigned: November 8, 2012 Due on: November 15, 2012

Instructions: This homework has 2 problems that can be solved in groups of size at most 3. Please follow the homework guidelines given on the class website; submitions not following these guidelines will not be graded.

Recommended Reading: Lectures 18 through 20. You may find it particularly useful to look at Lecture 20 proof of theorem 1 (pages 3 to 6) to see how you may describe a large Turing machine clearly.

Problem 1. [Category: Comprehension+Design+Proof] A *queue automaton* is like a pushdown automaton except that the stack is replaced by a *queue*, i.e., when a symbol is *enqueued* onto the queue, it is written at the left-end of the queue, and when a symbol is *dequeued* from the queue it is removed from the right-end of the queue. Like a PDA, in each step, depending on the current control state, the current input symbol read (which could be ϵ meaning nothing is read from the input), and the symbol dequeued from the queue (which could be ϵ, meaning that no symbol is dequeued), the control state at the next time instant and the symbol to be enqueued (which could be ϵ meaning that no symbol is enqueued) is determined. Additionally, like a PDA, the queue machine accepts an input if it reaches an accepting/final state after reading the entire input, irrespective of the contents of its queue.

1. Give the formal definition of a queue machine as a tuple, giving the domain and range of the transition function. **[2 points]**
2. Give the formal definitions of the instantaneous description of the queue machine, computation on a word, and the language accepted by the machine. **[3 points]**
3. Prove that if M is a (deterministic, single-tape) Turing machine then there is a queue machine P such that $\mathbf{L}(P) = \mathbf{L}(M)$. **[10 points]**

Aside: Queue automata arise naturally as models of distributed computing environments. Imagine a distributed systems where agents communicate with each other by sending and receiving messages over channels, and the channels themselves buffer the messages to ensure that message sends are non-blocking, and that messages are delivered in the order in which they were sent. If each agent in the distributed system has only finite memory, then the entire system can be modelled as a queue automata (with many queues), where the control state of the automaton is the cross-product of the state of each agent, and there is a queue associated with each communication channel which is the message buffer for that channel. What this problem says is that such a distributed system can "compute" anything that a general purpose computer can, even if each agent is a device with very limited computing ability (like is a finite automaton).

Problem 2. [Category: Comprehension+Design+Proof] Consider a variation of Turing machines: No-InpuT-WrIte-Turing machine (abbreviated as NITWIT). NITWIT is a deterministic TM with a single two-way infinite tape (i.e., there is no leftmost cell on the tape) with the restriction that it is not allowed to write on the input portion of the tape. Note that the machine is allowed to write anything it pleases outside the portion of the tape where the input is written. Additionally, a NITWIT machine enters the accept state q_{acc} or the reject state q_{rej} only when the head is within the input portion of the tape. In this problem, we will show that, as their name suggests, NITWIT machines are not as clever as Turing machines, and can only (surprise!) recognize regular languages.

Formal Definition of NITWIT: A NITWIT machine is not allowed to write on the input portion of the tape. So the machine will keep track (in its finite control) whether it is within the input portion of the tape or outside the input portion of the tape — within the input portion, it must write the same symbol it reads in each step, whereas outside the input portion it can write anything it chooses. Finally, we will assume that there is left-end-marker ($\rhd$) and a right-end-marker ($\lhd$) that mark the beginning and end of input portion of the tape; the machine is never allowed to write these symbols anywhere else on the tape. Formally, a NITWIT machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

- $Q = Q_{\text{in}} \cup Q_{\text{out}}$ is a finite set of states, where $Q_{\text{in}} \cap Q_{\text{out}} = \emptyset$. The machine will be in a Q_{in} state when the tape head is in the input portion of the tape, and is in state Q_{out} when the tape head is outside the input portion of the tape.
- Σ is the (finite) input alphabet containing two special symbols: $\rhd$, the left-end-marker, and $\lhd$, the right-end-marker. The input is assumed to be a string of the form $\rhd w \lhd$ where $w \in (\Sigma \setminus \{\rhd, \lhd\})^*$, i.e., w does not contain the symbols $\rhd$ and $\lhd$.
- $\Gamma \supseteq \Sigma$ is the tape alphabet with $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state.
- $q_{\mathsf{acc}} \in Q_{\text{in}}$ is the unique accept state. Since the machine accepts only when the head is within the input portion, q_{acc} must be in Q_{in}.
- $q_{\mathsf{rej}} \in Q_{\text{in}}$ is the unique reject state; again, q_{rej} must be in Q_{in} because the machine rejects only when the tape head is within the input.
- Finally $\delta : (Q \times \Gamma) \to (Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\})$ with the following restrictions.
  - When the machine is not scanning the end markers and the head is within the input then at the next step it remains within the input and it must write the same symbol. Formally, if $\delta(q_1, a) = (q_2, b, D)$ with $q_1 \in Q_{\text{in}}$ and $a \in \Gamma \setminus \{\rhd, \lhd\}$ then $q_2 \in Q_{\text{in}}$ and $b = a$.
  - When the machine is scanning the left-end-marker, and it moves right then it moves within the input. Formally, if $\delta(q_1, \rhd) = (q_2, b, \mathsf{R})$ then $b = \rhd$ and $q_2 \in Q_{\text{in}}$.
  - When the machine is scanning the left-end-marker, and it moves left then it moves outside the input. That is, if $\delta(q_1, \rhd) = (q_2, b, \mathsf{L})$ then $b = \rhd$ and $q_2 \in Q_{\text{out}}$.
  - When the machine is scanning the right-end-marker, and it moves left then it moves within the input. Formally, if $\delta(q_1, \lhd) = (q_2, b, \mathsf{L})$ then $b = \lhd$ and $q_2 \in Q_{\text{in}}$.
  - When the machine is scanning the right-end-marker, and it moves right then it moves outside the input. That is, if $\delta(q_1, \lhd) = (q_2, b, \mathsf{R})$ then $b = \lhd$ and $q_2 \in Q_{\text{out}}$.
  - Finally, if the head is outside the input portion then it remains outside the input in the next step. That is, if $\delta(q_1, a) = (q_2, b, D)$ with $q_1 \in Q_{\text{out}}$ and $a \in \Gamma \setminus \{\rhd, \lhd\}$ then $q_2 \in Q_{\text{out}}$ and $b \in \Gamma \setminus \{\rhd, \lhd\}$.

  Notice, when the machine is scanning the left-end-marker or right-end-marker, it could be in a state of Q_{in} or Q_{out} depending on whether it came from within the input or from outside the input.

Since NITWIT machines are special forms of Turing machines, definitions of configurations, one step, acceptance, and language recognized are exactly the same as those for Turing machines. The initial configuration of the machine on input $w \in (\Sigma \setminus \{\rhd, \lhd\})^*$ is $q_0 \rhd w \lhd$.

Computations of NITWIT: A computation of a NITWIT machine takes the following form: the machine enters the input portion of the tape from the left or the right end (initially it enters from the left end), executes a sequence of steps where the head is within the input portion, then reaches one of the end markers (or may never reach an end-marker if it goes into an infinite loop), then it executes a sequence of steps when the head is outside (this could be 0 steps or infinite, if the machine goes into an infinite loop), and then re-enters the input portion, and repeats this until it halts. Thus, a computation of NITWIT can be broken into "phases" where the machine "enters" the input portion of the tape from one of the end markers, and then "leaves" when it moves to one of the end-markers from within the input portion. Notice that during such a phase, the behavior of the machine only depends on the input (as the portion outside the input is not read) and the tape contents don't change. So to describe the effect of such a phase we don't need to consider the tape. We will define the following function to describe these phases. Let us fix a NITWIT machine M. For an input $w \in \Sigma^*$, let us define a function $f_w : Q \times \{\rhd, \lhd\} \to ((Q \times \{\rhd, \lhd\}) \cup \{A, R, \infty\})$. For $q, q' \in Q$, $E, E' \in \{\rhd, \lhd\}$, define $f_w(q, E)$ to be equal to

- (q', E') if when M enters in state q from E, it leaves the input portion from in state q' at end-marker E'; note that if $E = \rhd$ and $\delta(q, \rhd) = (q_1, \rhd, \mathsf{L})$, or if $E = \lhd$ and $\delta(q, \lhd) = (q_1, \lhd, \mathsf{R})$, then we take $f_w(q, E) = (q, E)$,
- A if M accepts within w (when M enters input w in state q at end-marker E)
- R if M rejects within w (when M enters input w in state q at end-marker E), and
- ∞ is M goes into an infinite loop and stays within the input, when M enters w in state q at end-marker E.

We now study the properties of the functions f_w defined above.

1. How many different functions f_w are there? In other words, what is the size of the set $\{f_w \mid w \in \Sigma^*\}$. **[2 points]**
2. Prove that if $f_{w_1} = f_{w_2}$ [1] then $w_1 \in \mathbf{L}(M)$ implies $w_2 \in \mathbf{L}(M)$. Notice that since the condition is symmetric on w_1, w_2, proving this shows that if $f_{w_1} = f_{w_2}$ then $w_1 \in \mathbf{L}(M)$ iff $w_2 \in \mathbf{L}(M)$. *Hint:* You may want to prove this by induction on the number of times the computation of M on w_1 enters/leaves the input portion. **[5 points]**
3. Prove that if $f_{w_1} = f_{w_2}$ then for all $w \in \Sigma^*$, $f_{w_1 w} = f_{w_2 w}$. **[5 points]**
4. Recall the equivalence $\equiv_{\mathbf{L}(M)}$ on strings defined as $u \equiv_{\mathbf{L}(M)} v$ when for all $w \in \Sigma^*$, $uw \in \mathbf{L}(M)$ iff $vw \in \mathbf{L}(M)$. Recall also the Myhill-Nerode theorem which says that $\mathbf{L}(M)$ is regular iff $\equiv_{\mathbf{L}(M)}$ has finitely many equivalene classes. Using the previous parts of this problem, prove that $\mathbf{L}(M)$ is regular. **[3 points]**

[1] Two functions $f, g : X \to Y$ are said to be equal, if for each $x \in X$, $f(x) = g(x)$.