CSCE 551 Midterm II Answers, July 28, 2014

1. (10 points) Fill in the blanks in the following proof via the Pumping Lemma that the language \( L := \{ w \in \{0, 1\}^* \mid w \) has more 0’s than 1’s \} is not regular:

   Given \( p > 0 \),
   let \( s := \) _______. Clearly, \( s \in L \) and \( |s| \geq p \).
   Given any \( x, y, z \) such that \( s = xyz \), \( |y| > 0 \), and \( |xy| \leq p \),
   let \( i := \) _______.
   Then \( xy^i z \notin L \), because [Explain briefly.] _______.

   Answer: We let \( s := 0^p1^{p-1} \) and let \( i := 0 \). By our choice of \( s \), we know that \( y \) must
   be of the form \( 0^r \) for some \( r > 0 \). So then, \( xy^0 z = xz = 0^{p-r}1^{p-1} \), but \( p - r \leq p - 1 \),
   which means that \( w \) does not have more 0’s than 1’s, whence \( xz \notin L \).
   Other answers are possible.

2. (20 points total)

   (a) (10 points) Describe formally with a transition diagram a Turing machine \( M \) with
   input alphabet \( \Sigma := \{0, 1, \#\} \) that makes a copy of a binary string. That is, on
   any input of the form \( w\# \) where \( w \in \{0, 1\}^* \), your machine \( M \) eventually halts,
   leaving \( w\#w \) on the left end of its tape (and blanks everywhere else).
   Please note that you can assume the input is \( w\# \) for some binary string \( w \), and
   so the separator symbol \( \# \) is already on the tape when the computation begins.
   If the input is not of this form, then \( M \) may behave arbitrarily.

   (b) (10 points) Give an implementation-level description of \( M \).

   Answer:

   (a) Shown below are two possibilities for \( M \). The TM on the left has 7 states (the
   reject state is not shown) and tape alphabet \( \{0, 1, \#, x, _\} \). The underscore \( (\_\) stands for the blank symbol. The TM on the right has 6 states and tape alphabet
   \( \{0, 1, \#, \bar{0}, \bar{1}, _\} \).
(b) Each TM above copies each symbol of $w$, from left to right, over to the right of the #, until it finishes with $w$ (encountering the # symbol). It does this by marking its current location in $w$, recording the binary symbol found there in its state, then scanning right and copying that symbol in the first blank cell found. It then scans left back to the marked location, restores the original symbol that was there, then shifts right to the next symbol, marks it, and so on.

3. (15 points) Assume some fixed alphabet $\Sigma$. Give a high-level description of a decision procedure for the following language:

$$L := \{ \langle R, S \rangle \mid R \text{ and } S \text{ are regexps and } L(R)L(S) = L(S)L(R) \}.$$  

(Note: $L(R)L(S)$ means the concatenation of language $L(R)$ with $L(S)$, i.e., the set of all strings of the form $xy$ such that string $x$ matches $R$ and string $y$ matches $S$.)

**Answer:**

"On input $\langle R, S \rangle$, where $R$ and $S$ are regular expressions:

(a) Construct two regular expressions $T_1 := RS$ and $T_2 := SR$.
(b) Construct two DFAs $A_1$ and $A_2$ equivalent to $T_1$ and $T_2$, respectively.
(c) Run the decider for $EQ_{DFA}$ (given in Chapter 4) on input $\langle A_1, A_2 \rangle$.”

(This decision procedure works because

$$L(R)L(S) = L(RS) = L(T_1) = L(A_1),$$

$$L(S)L(R) = L(SR) = L(T_2) = L(A_2).$$"
and thus \( \langle R, S \rangle \in L \) if and only if \( \langle A_1, A_2 \rangle \in EQ_{DFA} \).

4. (20 points) Let \( A \) be the language consisting of all strings \( \langle M, t \rangle \) such that

- \( M \) is a Turing machine,
- \( t \) is a natural number, and
- \( M \) never halts in fewer than \( t \) steps on any input.

Show that \( A \) is decidable by giving a high-level description of a decider for \( A \). [Hint: You only need to run \( M \) on a finite number of input strings. Which strings?]

**Answer:** The idea is that a machine can only look at the first \( t \) symbols of its input before \( t \) steps, so if it halts on any string at all, it will halt on a string of length less than or equal to \( t \). Thus we limit our search space to the strings of length \( \leq t \), of which there are only finitely many.

“On input \( \langle M, t \rangle \) where \( M \) is a TM and \( t \) a natural number:

(a) For all strings \( w \) such that \( |w| \leq t \), do
   i. Run \( M \) on input \( w \) for \( t \) steps.
   ii. If \( M \) halts on input \( w \) in fewer than \( t \) steps, then reject.

(b) Accept.”

5. (15 points) Let \( f : \Sigma^* \rightarrow \Sigma^* \) be any function, and define

\[
\text{range}(f) := \{ f(w) \mid w \in \Sigma^* \} .
\]

(Note that \( \text{range}(f) \) is a language over \( \Sigma \).) Show that if \( f \) is computable, then \( \text{range}(f) \) is Turing-recognizable. [Hint: you can describe either a recognizer of \( \text{range}(f) \) or an enumerator for \( \text{range}(f) \), as you wish.]

**Answer:** Here is an enumerator for \( \text{range}(f) \):

(a) Cycling through all strings \( w \), do the following for each \( w \):
   i. Compute \( z := f(w) \).
   ii. Print \( z \).

Here is a recognizer for \( \text{range}(f) \):

“On input \( w \), where \( w \) is any string:

(a) Cycling through all strings \( z \), do the following for each \( z \):
   i. Compute \( w' := f(z) \).
   ii. Check whether \( w = w' \):
A. if yes, then accept;
B. otherwise, continue on to the next $z$.”

6. (15 points EXTRA CREDIT) Suppose $A \subseteq \Sigma^*$ and $A \neq \emptyset$. Show that if $A$ is Turing-recognizable, then $A = \text{range}(f)$ for some computable function $f$. (Refer to the previous problem for a definition of $\text{range}(f)$.) [Hint: you can assume as given either a recognizer of $A$ or an enumerator for $A$, as you wish.]

**Answer:** Since $A \neq \emptyset$, we can fix some string $d_0 \in A$ (which we might call the “default string”). Given an enumerator $E$ for $A$, here is how we can compute a function $f$ whose range is $A$:

“On input $w$, where $w$ is any string:
(a) If $w$ is not of the form $\langle t \rangle$, where $t$ is a natural number, then output $d_0$.
   ($d_0$ is hard-coded into our TM computing $f$.)
(b) (Otherwise, $w = \langle t \rangle$ for some natural number $t$.) Extract $t$ from $w$.
(c) Run $E$ for $t$ steps.
(d) If $E$ does not print any string within $t$ steps, then output $d_0$.
(e) (Otherwise,) let $x$ be the last string printed by $E$ within $t$ steps.
(f) Output $x$.”

Here is a simpler algorithm (for a different $f$):

“On input $w$, where $w$ is any string:
(a) Run $E$ for $|w|$ many steps or until it prints something, whichever comes later.
(b) Output the last string printed by $E$, above.

Here is an algorithm for yet another $f$, given a recognizer $M$ for $A$:

“On input $w$, where $w$ is any string:
(a) If $w$ is not of the form $\langle x, t \rangle$, where $x$ is a string and $t$ is a natural number, then output $d_0$.
(b) (Otherwise, $w = \langle x, t \rangle$ as above.) Extract $x$ and $t$ from $w$.
(c) Run $M$ on input $x$ for $t$ steps.
(d) If $M$ accepts $x$ within $t$ steps, then output $x$.
(e) Otherwise, output $d_0$.

Other solutions are possible.