Harvard University
Computer Science 121

Problem Set 8

Due Tuesday, November 12, 2013 at 11:59 PM.
Submit your solutions electronically to cs121+ps8@seas.harvard.edu with "ps8 submission" in the subject line. The solutions to Parts A and B should be attached as separate PDF files, called lastname+ps8a.pdf and lastname+ps8b.pdf.
Late problem sets may be turned in until Friday, November 15, 2013 at 11:59 PM with a 20% penalty.

Problem set by **ENTER YOUR NAME HERE**

Collaboration Statement: **FILL IN YOUR COLLABORATION STATEMENT HERE (See the syllabus for information)**

See syllabus for collaboration policy.

PART B (Graded by Louis and Perry)

PROBLEM 1 (8+8 points)

Let $S = \{\langle M \rangle : M \text{ is a TM and } L(M) = \{\varepsilon\}\}$. By using mapping reductions:

(A) Show that S is not recognizable.

(B) Show that S is not co-recognizable.

(A) $\overline{HALT_{TM}} = \{< M, w >: \text{M does not halt on } w\}$.
We show that $\overline{HALT_{TM}} \leq_m S$.
Given an M and a w, define M' to be the TM that runs the following procedure:

- On input x, if $x = \varepsilon$, accept.
- Otherwise, run M on w, and if M halts, accept, and if not, then continue running M on w forever.

If M does not halt on w, then M' accepts only ε, since it simulates M forever on any other input. So $L(M) = \{\varepsilon\}$.

If M does halt on w, then M' always accepts. So $L(M) = \Sigma^*$.

Therefore, M does not halt on w iff $L(M') = \{\varepsilon\}$, and defining f to be the function such that $f(< M, w >) =< M' >$, we have that $x \in \overline{HALT_{TM}}$ if and only if $f(x) \in S$. Therefore, f is a reduction, and $\overline{HALT_{TM}} \leq_m S$.

Moreover, $\overline{HALT_{TM}}$ is not recognizable (because if it were, then $HALT_{TM}$ would be both recognizable and co-recognizable and therefore decidable). Since $\overline{HALT_{TM}}$ is not recognizable and $\overline{HALT_{TM}} \leq_m S$, S is not recognizable.

(Strictly speaking, $\overline{HALT_{TM}}$ also contains all the strings that aren't valid encodings of the form $< M, w >$. We can account for this by simply having having f map invalid encodings to some

arbitrary TM in S, such as the TM that accepts ε and immediately rejects if its input is anything else).

(B) We show that $HALT_{TM} \leq_m S$. We apply the same argument as in part B, except that we define M' slightly differently.

Given an M and a w, define M' to be the TM that runs the following procedure:

- On input x, run M on w.
- If M halts, accept if $x = \varepsilon$ and reject otherwise. If M doesn't halt, then continue running M on w forever.

If M halts on w, then M' accepts iff $x = \varepsilon$. So $L(M) = \{\varepsilon\}$.

If M does not halt on w, then M' always runs forever, so it accepts nothing. So $L(M) = \emptyset$.

Therefore, M halts on w iff $L(M') = \{\varepsilon\}$, and defining f to be the function such that $f(< M, w >) =< M' >$, we have that $x \in HALT_{TM}$ if and only if $f(x) \in S$. Therefore, f is a reduction, and $HALT_{TM} \leq_m S$.

Moreover, $HALT_{TM}$ is not co-recognizable (because if it were, then $HALT_{TM}$ would be both recognizable and co-recognizable and therefore decidable). Since $HALT_{TM}$ is not co-recognizable and $HALT_{TM} \leq_m S$, S is not co-recognizable.

PROBLEM 2 (8 points)

Let a "rep-string" be a string of the form $w = xx$ for $x \in \Sigma^*$.
Consider the language $L = \{\langle G \rangle : G \text{ is a CFG and } L(G) \text{ contains a rep-string}\}$.
Prove that L is undecidable.

Solution:
Remember that we showed in lecture that the language $NULL = \{\langle G_1, G_2 \rangle : L(G_1) \cap L(G_2) = \emptyset\}$ is undecidable for CFGs G_1 and G_2. We will show that $NULL \leq_m L$ so L is undecidable.

Let # be a fresh terminal not found in either G_1 or G_2. Let the f in the mapping reduction compute as follows: Given input $\langle G_1, G_2 \rangle$, construct a new CFG G_f where $L(G_f) = L(G_1) \circ \# \circ L(G_2) \circ \#$ by taking the start states S_1, S_2 for G_1, G_2, and letting G_f have all of the rules G_1, G_2 have but a new start state S with $S \to S_1 \# S_2 \#$. $\langle G_f \rangle$ is the output, so $f(\langle G_1, G_2 \rangle) = \langle G_f \rangle$.

A rep-string $w \in L(G_f)$ must be of the form $w_1 \# w_2 \#$ with $w_1 \in L(G_1)$, $w_2 \in L(G_2)$, and $w_1 = w_2$. Thus if $L(G_1) \cap L(G_2) = \emptyset$, then $L(G_f)$ has no rep-strings. Similarly, if $L(G_1) \cap L(G_2) \neq \emptyset$ then there is a $w_1 \in L(G_1), L(G_2)$, so $w_1 \# w_1 \# \in L(G_f)$ and $L(G_f)$ has a rep-string.

Thus $\langle G_1, G_2 \rangle \in NULL$ iff $\langle G_f \rangle \in L$, so f is a computable function such that $\forall w \in \Sigma^*, w \in NULL$ iff $f(w) \in L$ and we are done.

PROBLEM 3 (4+4 points)

In the near future you're working as an engineer, when your manager asks you to write the following two programs. Is this a problem? Why or why not?

(A) Take another program's code as input and decide if that program is implemented in the fewest possible lines of code.

(B) Take another program's code and remove all inaccessible (dead) code from it.

(A) Turing machines can simulate any computation and minimizing a TM is undecidable, so the program cannot solve all cases.

Proof by contradiction. We assume this TM is decidable and demonstrate how using it we could decide ATM. We take the TM of interest and modify it so it erases its input and writes a new string of interest. It then runs as normal. This machine never alters its output—it is a constant function—so its minimal version is either an accepting or rejecting state. If the state is accepting then the original TM would have accepted, and if rejecting it would have rejected. This allows us to decide ATM, an undecidable language, a contradiction.

(B) Turing machines can simulate any computation and removing dead states from a TM is undecidable, so the program cannot solve all cases.

The proof is nearly identical to the above, except the dead states will be all off path states for a computation on a particular string. If the remaining machine, the input's path through the original, includes the halt and accept state we accept, otherwise reject.