Jack Conger
5/04/2015
Assignment #4

1. Let $J = \{w | w = 0x \text{ for some } x \in A_{TM} \text{ or } w = 1y \text{ for some } y \in \overline{A_{TM}}\}$. Show that neither J nor $\bar{J}$ is Turing-recognizable.

   If either J or $\bar{J}$ were recognizable with recognizer M or $\overline{M}$, respectively, then A_{TM} would be decidable, by reduction, with a decider as follows. First, we use M:

   - On input $\langle M \rangle, w$, where w describes a Turing machine and input:
   - On separate tapes, run M on $0w$ and $1w$ until M accepts on one of these.
   - If M accepted on $0w$, accept. If M accepted on $1w$, reject.

   To reduce using $\overline{M}$, we use the related machine:

   - On input $\langle \overline{M} \rangle, w$, where w describes a Turing machine and input:
   - On separate tapes, run $\overline{M}$ on $0w$ and $1w$ until $\overline{M}$ accepts on one of these.
   - If $\overline{M}$ accepted on $0w$, reject. If $\overline{M}$ accepted on $1w$, accept.

   We must show step 2 of each of these machines is guaranteed to run in finite time. For the first machine, we know w must either be in A_{TM} or $\overline{A_{TM}}$, since these partition the set of w which describe a Turing machine and input. Because of this, M must stop and accept on either $0w$, if w is in A_{TM}, or $1w$, if w is not in A_{TM}. If we run these two simulations simultaneously on two tapes only waiting until one accepts, which one must, this step will always halt.

   The reduction using $\overline{M}$ is very similar. On any words $0w$ or $1w$ with w appropriately formed, $\overline{M}$ will recognize $0w$ exactly when w is *not* in A_{TM} and $1w$ exactly when w *is* in A_{TM}. The proof of halting is then the same as the last paragraph.

2. Show that there is an undecidable language contained in 1^*.

   There are a countable number of Turing machines, which means we can index TMs on the natural numbers $0, 1, 2, \ldots$. These are exactly the numbers representable by 1^*. Represented this way, $\langle E_{TM} \rangle$ is a subset of 1^*, and therefore we have undecidable subsets of 1^*.

3. Which of the following languages are decidable? Justify each answer:

   (a) Given a Turing machine M, does M accept 0101?

   This is decidable, with the machine T as follows:

   - On input M,
   - Run M on the letters of 0101.
   - If M accepts by the end of 0101, accept. Else reject.

   This will run in finite, and in fact rather short, time, taking the time to simulate four steps on an input machine.

   (b) Given Turing machines M and N, is $L(N)$ the complement of $L(M)$?

   This is, in general, undecidable; one would need to test every word in Σ^* to verify that $L(N)$ accepts and $L(M)$ rejects that word, or vice versa, which cannot be done in finite time.

   (c) Given a turing machine M, integers a and b and an input x, does M run for more than $a|x|^2 + b$ steps on input x?

   This is decidable, with a decider T:

- On input M, a, b, x:
- Simulate M on x for at most $a|x|^2 + b$ steps.
- If M has halted before or on this number of steps, reject. Otherwise accept.

This will happen in at most $a|x|^2 + b$ simulation steps, and so will run in finite time.

4. Prove that if K and L are decidable by Turing machines running in polynomial time then so are $K \cup L$, KL, and $\overline{L}$.

If L is decidable by a machine M, then $\overline{L}$ is decidable by a machine M', which accepts exactly when M rejects, and rejects exactly when M accepts, and is otherwise identical. M' will run in exactly the same time as M, so if M runs in polynomial time, so does M'.

If K and L are both decidable by polynomial-time Turing machines M_K and M_L, then their union is decidable by a machine M, which runs as follows:

- On input x:
- Copy x after the right end of x.
- Run as M_K on x. In any case M_K accepts, accept.
- In any case M_K rejects, write the copy of x back to the left end of the tape. Then, run as M_L. In any case M_L accepts, accept. In any case M_L rejects, reject.

M will run in, at most, the time it takes to copy x, the time it takes to run M_K, the time it takes to copy x back, and the time it takes to run M_L. M_K and M_L run in polynomial time, and steps 1 and 3 run in no more than $O(|x|^2)$ each; thus M runs in polynomial time.

Lastly, if K and L are decidable in polynomial time, then KL is as well, by a machine that runs as follows:

- For all input strings x:
- We first store x on one tape, and partition and run on two other tapes, each where we put each sub-word of x.
- With this, for each partition of x into two sub-words x_1, x_2:
- Run M_K on x_1, M_L on x_2. If both M_K and M_L accept, then accept. Otherwise, move on to the next partition of x.
- If all partitions of x have been tried and have not accepted, then reject.

The act of repartitioning x takes $O(|x|)$ time, and each M_K and M_L run is in polynomial time. We can simulate this machine on one tape, amplifying this time, giving us a total time of $(O(|x|) + \max(T_K, T_L))^2$, which is polynomial.

5. Let $TRI = \{\langle G\rangle | G$ is an undirected graph that contains a triangle$\}$. Prove that there is a polynomial-time Turing machine that decides TRI.

Enumerate all triples of nodes $(a, b, c) \in G$. This takes $O(n^3)$ time, where n is the number of nodes in G. For each triple, check if all edges (a, b), (a, c), (b, c) exist in G. This second step takes $O(e)$, where e is the number of edges in G. Thus the running time is $O(n^3) + O(n^3 e)$, which is polynomial in the length of $\langle G\rangle$.