CS 182, Complexity Theory
Fall 2010
Homework 2
Due Wednesday, September 15

1. **[10 Points] The Recursion Theorem Revisited.** Your task in this problem is to use the Recursion Theorem to prove that each of the following languages is not recursive. (We know how to show this using Rice's Theorem or similar reduction techniques, but here the point is to do the proofs using the Recursion Theorem.)

   (a) $L_{recursive} = \{< M > | M \text{ is a TM and } \mathcal{L}(M) \text{ is recursive}\}$.

   (b) $L_{tape} = \{< M, w > | \ M \text{ uses a finite amount of tape when run on } w\}$.

2. **[15 Points] Proving Rice's Theorem using the Recursion Theorem!**

   Recall that Rice's Theorem states that "Every non-trivial property of the recursively enumerable languages is undecidable." We proved this theorem by a general reduction from the halting problem. It is highly recommended that you review this "standard" proof of Rice's Theorem before you embark on this problem, both to build up your reduction muscles and to ensure that you feel comfortable with the notion of a non-trivial property. You are welcome to come and talk to Ran about this if you'd like.

   Your task here is to prove Rice's Theorem again, this time using the Recursion Theorem rather than using a reduction from a known undecidable problem.

3. **[20 Points] LTM's: The "Limit" of Computation!**

   In this problem we'll investigate a class of languages richer than the recursively enumerable languages but not as rich as the set of all languages.

   A *Limiting Turing Machine*, referred to henceforth as an LTM, is a Turing Machine that never halts on any input w. Instead, after each step of computation, an LTM writes the symbol "Y" (standing for "Yes") or "N" (standing for "No") on a special write-only (no reading from this tape head) right-only (the tape head can only be moved to the right) output tape indicating whether the TM "currently believes" that w is in the language. Thus, an LTM may "change its mind" any number of times (even an infinite number of times) as to whether the word should be accepted. The LTM, of course, has other regular tapes on which it can write and move both right and left.

If M is an LTM and w is a word, we say that M *accepts w in the limit* if there exists some point in the computation beyond which M always writes "Y" on the tape at each subsequent computation step. Conversely, we say that M *rejects w in the limit* if past some point in the computation M writes "N" at each subsequent computation step. If neither of these things happen (that is, M changes its mind an infinite number of times) then M is said to have *undefined behavior* on input w. Finally, we say that M *decides a language L in the limit* if for all inputs $w \in \Sigma^*$, M accepts w in the limit if $w \in L$, and M rejects w in the limit if $w \notin L$. (Thus, M does not have undefined behavior on any input string.)

(a) Show that every recursively enumerable language L has an LTM M that decides L in the limit. (Thus, you are showing that LTM's are at least as powerful as TM's.)

(b) Show that there exists a language L that is *not* recursively enumerable such that L is decided in the limit by some LTM.

(c) Prove that there still exists a language which is not decided by any LTM in the limit.

4. **[15 Points] Closure under Kleene Star!** Recall that if L is a language then $L^0 = \{\epsilon\}$, $L^1 = L$, and $L^i = LL^{i-1}$ (the concatenation of L and L^{i-1}). The Kleene Closure (pronounce "Clean Knee")[1] of L is denoted L^* and is defined to be

$$\bigcup_{i \geq 0} L^i$$

In other words, L^* is the language made up of all strings w where w can be written as the concatenation of 0 or more strings in L.

[1]From the History of Mathematics Archives at University of St. Andrews, Scotland: "Stephen C. Kleene studied for his first degree at Amherst College. He went on to receive a doctorate from Princeton University in 1934, supervised by Church, for a thesis entitled "A Theory of Positive Integers in Formal Logic." Then Kleene taught at Princeton until he joined the University of Wisconsin at Madison in 1935. He became a full professor at the University of Wisconsin at Madison in 1948 and remained on the staff there until he retired in 1979. Kleene's research was on the theory of algorithms and recursive functions. He developed the field of recursion theory with Church, Godel, Turing and others. He contributed to mathematical Intuitionism which had been founded by Brouwer. His work on recursion theory helped to provide the foundations of theoretical computer science. By providing methods of determining which problems are soluble, Kleene's work led to the study of which functions can be computed."

(a) (Warmup) Show that if L is recursive then L^* is recursive.

(b) (Warmup) Show that if L is recursively enumerable then L^* is recursively enumerable.

(c) (Moderate) Show that if L is in NP then L^* is in NP.

(d) (A bit more challenging) Show that if L is in P then L^* is in P.

5. **[40 Points] A Self-Printing Program!** Write a self-printing program in python, C++, or Java in the spirit of the proof of the Recursion Theorem that we saw in class. (There are other ways to write self-printing programs, but we want a recursion theoretic solution.) When run, your program should print out precisely its own code. Most computers don't have infinite tapes, so you'll need to use something else to simulate the tape. An array, a string, or a list can work very well for this purpose. *If you find yourself bumping your head against the wall, please feel free to come and visit me for a modest suggestion that will get you rolling in the right direction.*

   Your submission should include the following:

   - A printout of the code.
   - Circle the part of the code that corresponds to "A" in the Recursion Theorem proof and the part of the code that corresponds to "B" in that proof.
   - A printout of the program running. A nice way to do this is using `script` on a Unix machine. (See the man pages.)

6. **[20 Point OPTIONAL BONUS PROBLEM] Every TM has a corresponding unrestricted grammar.** In class we stated, without proof, that the languages generated by unrestricted grammars are exactly the recursively enumerable languages. Prove it!