Comments on Homework 4

The big ideas and common problems

1. (5 pts) problem 10.5 (If a finite set of recursively enumerable languages partition $\Sigma^*$, then the languages are recursive).

Everyone got this one – the conditions imply that each language’s complement is also recursively enumerable, and since the language and its complement are recursively enumerable the language is recursive. However, everyone should realize the difference between this problem and the problem on the last homework (infinite unions of recursive languages). Recursive and recursively enumerable languages are closed under finite unions, but not infinite unions.

2. (5 pts) Problem 10.23 on page 309 (Let $S$ be the set of languages containing those $L \subseteq \{0,1\}^*$ such that neither $L$ nor its complement is recursively enumerable. Show that $S$ is uncountable, and give an example language in $S$.)

It is not too hard to show that the there are infinitely many languages $L$ such both $L$ and the complement $\overline{L}$ are not recursively enumerable. Since there are only countably many Turing Machines, there are only countably many recursively enumerable languages. Similarly, there are only countably many languages who complements are recursively enumerable (there is the mapping from languages to their complements is a bijection, so there are as many recursively enumerable languages as there are languages whose complements are recursively enumerable). Note that this counts each recursive language twice, but that is OK. Starting with the set of all languages is uncountable and removing a countable set from an uncountable set leaves an uncountable set (twice) completes the argument.

To define a particular language $L$ that is not recursively enumerable and whose compliment is also not recursively enumerable was more difficult for the class. This uses a diagonalization argument where two strings are assigned to each Turing machine $T$ – perhaps $e(T)$ followed by a 0 and $e(T)$ followed by a 1. One string (say the $e(T)$1 string) is in the language $L$ if and only if $T$ accepts it, and the other is in $L$ if and only
if $T$ does not accept it. Those strings that are not the encoding of a TM followed by a zero or a one are (arbitrarily) defined to be in the language.

Key points: every string must be either in $L$ or not in order to define the language, every TM $T$ must have a string verifying that $T$ does not accept the language ($e(T)0$ in this case), and every TM $T$ must have a string verifying that $T$ does not accept the complement of $L$ ($e(T)1$ in this case).

3. (6 pts) Problem 10.31 on page 310 (show that if $L$ is any infinite recursively enumerable language then there is an $L' \subseteq L$ such that $L'$ is infinite and recursive). This one is tricky, see problem 10.30 for some things to think about.

If $L$ is an infinite recursively enumerable language then there is a Turing Machine $T$ that enumerates $L$. Here is the high level idea:

Create a new Turing Machine $T'$ from $T$ that enumerates some of the strings in $L$. Keep track of the last string enumerated by $T'$, and output the next string enumerated by $T$ if and only if it follows the string previously enumerated by $T'$ in canonical order. Since $L$ is infinite, and there are only finitely many strings before any particular string in canonical order, $T$ will eventually enumerate a string later (in canonical order) than the last string output by $T'$. Therefore $T'$ will output an infinite set of strings in canonical order, so the language enumerated by $T'$ is infinite and recursive (since it can be enumerated by a TM in canonical order).

Recall that canonical order when the alphabet is $\{0, 1\}$ is the one ordering of strings where shorter strings are listed before longer ones, and within each length strings are sorted as if they were binary numbers. This can easily be generalized to any other alphabet $\Sigma = \{a_0, a_1, \ldots, a_n\}$ by ordering the strings by length, and within each length as if they were $n$-arry numbers, so $a_0a_0a_0$ would be the first length 3 string, followed by $a_0a_0a_1$ and so on up to $a_na_na_n$. Thus listing any (finite) alphabet gives a canonical ordering of the strings over that alphabet.

4. (4 pts) Describe the languages generated by the grammars of exercise 11.1 (b) and (c).

The first generates strings of $a$’s such that the length is of the form $2^i3^j$ for some $i, j \in \mathbb{N}$.
The second generates strings in \( \{a, b, c\}^+ \) such that the number of a’s, b’s and c’s in the string are all the same.