Homework 4

Austin Frownfelter Matthew Bialecki

January 22, 2018

1 Problem 5

Show that the following two definitions of recursively enumerable are logically equivalent:

(a) A language is recursively enumerable iff there is a Turing machine M such that x∈L then M accepts x and if x∉L then M loops forever on x.

(b) A language is recursively enumerable iff there is a Turing machine M such that x∈L then M, with a read/write tape that is initially empty and a write-only output tape, such that only elements of L are written to the output tape, and every element of L is eventually written to the output tape.

Assume there is a Turing machine A which accepts input x if x∈L and loops forever if x∉L. There is a Turing machine B which will enumerate all strings y such that y∈L. B is defined as follows:

1. Nondeterministically generate all possible strings over the tape language.
2. For each string, run A with the string w$_i$ as input, with the following practice:
   a. Run A on w$_i$ for a single computation.
   b. Save the computation history of A on w$_i$.
   c. Run A on w$_{i-j}$ $\forall j \mid 0 < j \leq i$, continuing from the previous computation for that string.
   d. When any w$_x$ is accepted by A, output it to the output tape of B.

B will run A on all strings it generates. A will not halt if the string is not within the language. B will only run A for a single computation for each previous string generated. If a string is in the language, A will eventually accept it, so B will eventually print it out. If a string is not in the language, then A will loop forever, however B will be able to continue for the other strings it generates. Therefore, B will eventually enumerate all strings x such that x∈L. Since A accepts all strings only within the language L and B will generate all strings only within the language L, both machines, and therefore statements a and b, are logically equivalent.

2 Problem 6

Consider a proof rule such as:

Proof Rule: From the statement ∀x P(x) one can deduce the countably infinite number of statements P(0), P(1), P(2), P(3), etc.

Somewhat more formally, you can assume that there is a Turing machine T that takes as input a statement S, and will output a list all the statements one can conclude from S using this proof rule. Here ?list? means what it means within the context of the definition of recursively enumerable, T will only output statements we can deduce from S by the proof rule, and every statement that we can deduce from S by the proof rule will eventually be listed. Prove that the following language is still recursively enumerable:

L={S | statement S is provable from the finite set A of axioms}

Proof

Assume there exists a Turing machine M which accepts in input statement S and set of axioms A, then accepts if S is provable from A, and rejects or loops forever otherwise.

There exists a Turing machine G which generates all sets of axioms. There is a power set table such that the row indices represent the cardinality of the sets in that row. Since the number of axioms is infinite, the number of columns is infinite, and the number of rows is also infinite. Each row is the set of all sets of cardinality of that row's index. Consider the example of row 2. There is a two-dimensional table which represents all sets of cardinality 2, such that the row index represents the first axiom and the column index represents the second axiom. By traversing the table diagonally, it is possible to iterate over all possible sets in this table. This traversal can be placed linearly in the 2nd row of the power set table.

This can be generalized to an n-th row, with an n-dimensional table which can be traversed in such a way that every set is eventually reached. This power set table will therefore contain all possible sets of axioms. This power set can be traversed in a similar, diagonal manner, as to reach all elements. Therefore, G will eventually define all possible sets of axioms.

There exists a Turing machine Q which generates all statements S such that S∈L. Statements are strings such that they are valid statements. Q is defined as follows:

1. Nondeterministically generate all possible strings over the tape language.
2. For each string, do the following:
   a Nondeterministically use TM G to generate all sets of axioms.
   b For each set of axioms, run M with the set a_i as input, with the following practice:

i. Run M on a$_i$ for a single computation.
ii. Save the computation history of M on a$_i$.
iii. Run M on a$_{i-j}$ $\forall j \mid 0 < j \leq i$, continuing from the previous computation for that string.
iv. When any a$_x$ is accepted by M, output it to the output tape of Q.
v. Do the same thing as ii-iv but for the computation history of Q.

Q will run M on all statements and sets of axioms it generates. M will not necessarily halt if the statement is not within the language. Q will only run A for a single computation for each previous set generated. If the statement is in the language, then M will eventually accept it, so Q will eventually print it out. If a statement is not in the language, then that computation history will potentially loop forever. However, Q switches between all statements for single computations each, so the other statements will get the chance to accept. Therefore, Q will eventually generate all statements S∈L. Since Q recursively enumerates the language L, L is recursively enumerable.