CS 438/2404
Computability and Logic
ASSIGNMENT # 3
Due: 3pm, November 11, 2019

1. Prove that a theory Σ is consistent if and only if Σ has a model.

   Solution: Remember that a theory is a set of sentences closed under logical consequence, and a theory is consistent iff some sentence in the language is not in the theory.

   If Σ has a model, then Σ is consistent: Let $M \models \Sigma$. Let φ be a sentence in the language of Σ. If $M \models \varphi$ then $M \not\models \neg\varphi$ and $\neg\varphi \notin \Sigma$. If $M \not\models \varphi$, then $\varphi \notin \Sigma$. In both cases Σ is consistent.

   If Σ is consistent, then Σ has a model: Let Σ be a consistent theory, then there is a sentence in the language of Σ such that $\varphi \notin \Sigma$. Since Σ is a theory, we have $\Sigma \not\models \varphi$. But this means that there is a structure M such that $M \models \Sigma$ but $M \not\models \varphi$. This M is a model of Σ.

2. (10 points) Prove that a unary function f is recursive iff graph(f) is r.e. (Recall graph(f) is the relation $R(x, y) = (y = f(x))$. Note that f may not be total.

   Solution (sketch): For the direction $\Rightarrow$, suppose that f is recursive. Then some program $\{e\}$ computes f. Thus

   $$y = f(x) \Leftrightarrow \exists z(T(e, x, z) \wedge y = U(z))$$

   The RHS fits the definition of an r.e. relation. Alternatively we can consider a TM M that takes as input (x, y) and runs e on x. If the simulation halts and outputs y then M halts and accepts.

   Conversely, suppose that graph(f) is r.e. Then there is a recursive relation R such that

   $$y = f(x) \Leftrightarrow \exists z R(x, y, z)$$

   Let M_R be the Turing machine for R (that always halts and for a triple x, y, z, M_R on (x, y, z) accepts if $R(x, y, z) = 1$, and otherwise M_R halts and rejects.) Our TM M for computing f is as follows. Let $y_1, y_2, \ldots$ be an enumeration of all numbers, and similarly let $z_1, z_2, \ldots$ be an enumeration of all numbers. Then let $q_1, q_2, \ldots$ be an enumeration of all pairs (y_i, z_j). (For example, we could first enumerate all pairs of natural numbers whose sum is 0, and then enumerate all pairs of natural numbers whose sum is 1, etc.) On input x, during phase i M will simulate M_R on (x, q_i). If M_R halts and accepts, then M halts and outputs the first number in the pair q_i. Otherwise, M continues to the next phase. For any input x where f is defined, the above procedure will eventually halt and output $f(x)$, and thus f is recursive.

3. Are each of the following languages (i) recursive, (ii) r.e. but not recursive, (iii) not r.e. Prove your answer. Do not use the S-m-n theorem.

(a.) Let $\mathcal{L}$ be the set of all numbers x such that x codes a TM program, and 10 is in the range of the function computed by the program.

Solution: This language is r.e. but not recursive. We use dovetailing to show that it is r.e. Fix an enumeration $a_1, a_2, \ldots$ of all inputs. Tor $i = 1, 2, ...$: Simulate $\{x\}_1$ on the inputs $a_1, \ldots, a_i$ for i steps each. If any of the simulations halts and outputs 10, then halt and accept. Note that if 10 is in the range of $\{x\}_1$, then there is a minimal pair (a_j, t_j) such that $\{x\}_1$ halts and outputs 10 on a_j after t_j steps. Therefore our simulation will accept when in the i^{th} step of the loop, $i = max(j, t_j)$. If 10 is not in the range of $\{x\}_1$, our simulation will run forever and thus never accept x.

To see that it is not recursive, we will reduce K to $\mathcal{L}$. Given an input x to K, we modify x to obtain x' where the Turing machine $\{x'\}_1$ behaves as follows: it ignores its input and simulates $\{x\}_1$ on x; if $\{x\}$ halts on x then we halt and output 10. Now we claim that $x' \in \mathcal{L}$ if and only if $\{x\}_1$ halts on x: since $\{x'\}_1$ ignores its input, if $\{x\}_1$ halts on x, then $\{x'\}_1$ halts and outputs 10 on all of its inputs, and otherwise $\{x'\}_1$ doesn't halt on all of its inputs. Thus $\{x\}_1$ halts on x if and only if 10 is in the range of $\{x'\}$.) Since K is not recursive, $\mathcal{L}$ is also not recursive.

(b.) Let $\mathcal{L}$ be the set of all numbers x such that x encodes a TM program, and where the program coded by x halts on only finitely many inputs.

Solution: This language is not r.e. Recall that $K(y)$ accepts y whenever $\{y\}$ halts on input y. K is r.e. but not recursive, and thus K^c is not r.e. We will prove that L is not r.e. by showing $K^c \leq L$; that is, we will show that if L is r.e., then K^c is also r.e. Suppose for sake of contradiction that Q is an algorithm for L. That is, Q on input x accepts if $\{x\}$ halts on only finitely many inputs, and otherwise Q either rejects or gets into an infinite loop. We will use Q to construct an algorithm for K^c as follows. K^c on input y constructs the encoding, y' of an intermediate machine, where $\{y'\}_1$ on its input z behaves as follows. $\{y'\}_1$ simulates $\{y\}$ on input y for z time steps. If the simulation halts, then $\{y'\}$ goes into an infinite loop. Otherwise, $\{y'\}$ halts and accepts z. The algorithm for K^c calls Q on y' and accepts y if and only if Q accepts.

4. (5 points) Let $\mathcal{L}$ be a first order language with finitely many function symbols and predicate symbols. Prove that the set of unsatisfiable $\mathcal{L}$ sentences is recursively enumerable.

Solution: We use the completeness theorem. We can enumerate all LK proofs over $\mathcal{L}$. Given some formula A in $\mathcal{L}$, we enumerate through all LK proofs, and for each one, if it is a proof of the sequent $A \rightarrow$ then we halt and say that A is unsatisfiable.