ASSIGNMENT 5: TURING AND CHURCH

MICHAEL BEESON

For the following problems, we work with an unspecified "model of computation". That means we assume we have a computable partial function *App* such that $App(e, x)$ is the output of program e at input x, if and only if there is any such output. We write $App(e, x)\downarrow$ to mean that $App(e, x)$ is defined. We also assume that computations proceed in "steps", so there is a computable T-predicate as explained in lecture.

A "problem" is a predicate on the natural numbers $\mathbb{N}$. The problem P is said to be "computable" if its representing function is computable. In this terminology the halting problem is given by $P(e) \leftrightarrow App(e, e)\downarrow$. The diagonal method shows this P is not computable. A problem is called "unsolvable" if its representing function is not computable.

We define a problem Q to be *reducible* to another problem R if there is a total computable function f such that

$$Q(x) \leftrightarrow R(f(x))$$

1. Show that if an unsolvable problem Q is reducible to problem R, then R is also unsolvable.

2. The generalized halting problem, whether program e halts at input x, is represented formally as the set of pairs $\langle e, x\rangle$ such that $App(e, x)\downarrow$. Show that this problem is unsolvable, by reducing it to the problem P given above.

3. One of the following two sets of integers is easily shown to be computable. Which one? Explain why it is computable. What is the difficulty about deciding whether the other one is computable? (You are not asked to say whether the other one is computable, only why it is not obviously computable.)

(a) The set of integers d such that $a^2 + b^2 + c^2 = d$ has a solution in integers a, b, c.

(c) The set of integers d such that $a^3 + b^3 + c^3 = d$ has a solution in (positive or negative!) integers a, b, c. For example, when $d = 29$ a solution is $(3, 1, 1)$, and when $d = 30$ the smallest solution is $(-283059965, -2218888517, 2220422932)$.

4. Turing's original paper introduced the concept of a computable real number. He defined a computable real number to be one whose decimal expansion is given by a computable function. This definition is unsatisfactory, because $0.49999\ldots = 0.5$. Explain this remark by showing that with this definition, there will be problems with proving that the sum of two computable numbers is computable. (What is the sum of $0.49999\ldots + 0.000000\ldots$?) You are not required to *prove* that with this definition, the addition function won't be computable. Just explain why you can't give any obvious algorithm for computing the sum of two computable numbers.

5. Turing's definition can easily be fixed. First, we define a nonnegative rational number a/b to be an integer $\langle a, b\rangle$ with $b > 0$ and a and b having no common factor. Then we define the nonnegative real number x to be computable if there is a computable sequence of rationals r_n such that

$$|r - r_n| \leq 2^{-n}.$$

With this definition, prove that the sum and product of two computable numbers are computable numbers.

6. Prove that the numbers e and π are computable. (Hint, you can look up infinite series for those numbers. How can I compute π to within 2^{-n} and be sure I have that accuracy?)

7. Prove that there exists a non-computable positive real number. (Hint, use Cantor's diagonal method.)

8. (E. Specker 1949). Exhibit a bounded increasing sequence of computable real numbers x_k whose least upper bound is not computable. Hint: The n-th bit of the binary expansion of x will be 1 if and only if $App(n, n)\downarrow$. Arrange matters so that x_k represents the results of trying for k steps to compute the binary digits of x. To write your proof down, please use the T-predicate and the U-function from the lecture. The point of this exercise is to express an argument about "steps of a computation" using the T-predicate and the U-function.