ASSIGNMENT 8: COMPUTATIONS BY TURING MACHINES

MICHAEL BEESON

1. Recall that in a previous assignment you wrote a Turing machine to compute successor in binary notation. Using the simulator, find a string representation of its computation on input `111011`. That is, prepare a rectangular array in which each row shows the tape, with a vertical line marking the left of the input, and to the left of that line, column 0 shows the current state. Indicate the head position in each row somehow, e.g. by red highlighting, or a carat over the scanned square, or something.

2. Now move towards the official version of computation, as follows. According to the official definition, each tape square is a 16-bit integer. Ascii code of 0 is 48, ascii code of 1 is 49. Write out the list of six integers corresponding to the six tape squares on the first row of the computation; use ordinary decimal notation for the integers, separating them by commas. Don't forget the head-present flag. To the beginning of that string, add a 0 for the start state, and a comma.

3. That sequence of 7 integers has to be coded as a single integer to make an ITD.

(a) Using the method of coding sequences using powers of primes given in Lecture 4 to encode this sequence, write an expression involving powers of primes that represents the encoded sequence. (Do not attempt to evaluate that expression.)

(b) Observe that if it *were* evaluated, and you had to decode the sequence, you would have to start by factoring that large number. Factoring is famously difficult, but in this case (and generally for decoding sequences) it wouldn't be so bad. Why not?

4. Let φ_e be the partial function computed by Turing machine e. Use the **T**-predicate and the U-function to express $\varphi_e(x) \cong 1$ without using φ and without English.

5. The "number of steps" taken in a Turing machine computation is the number of machine instructions executed. Define $f(e)$ to be the number of steps taken by Turing machine e in computing $\varphi_e(0)$ (if that is defined). Use the **T**-predicate to show that f is a Turing computable function.

6. Turing machines have "unbounded memory" in the form of a (potentially) infinite tape. This exercise considers what happens if we restrict the machine to use only a limited amount of the tape, say ten times the length of the input string. When a move-right instruction would cause that bound to be violated, the machine halts; otherwise it works like a Turing machine. Let ψ_e be the function computed by Turing machine e with this modification. Show that the halting problem for such computations is solvable. More precisely, show that the set of e such that $\psi_e(e)$ halts is Turing computable (it is not

claimed that it is computable by such a restricted machine, but by an ordinary one). Hint: Show that, given the machine e and the input x, there are only finitely many possible ITD's. How many? What happens if an earlier ITD is repeated?