CS 81: Computability and Logic, Spring 2011

Assignment 10: Turing Machines

Due: Wednesday, April 20

- Emails about this assignment should be directed to cs81help@cs.hmc.edu .
- Grutor office hours in Platt are Sundays 8–10pm, Mondays 7–11pm, and Tuesdays 7pm-12m.
- Prof. Stone's official office hours in Olin 1251 are MW 4–5pm and TR 3–4pm, and Prof. Keller's official office hours in Olin 1253 are MTW 4-5:30pm; you can often catch us at other times (or make an appointment).
- Make sure your submission includes your name!

---

1. Read Chapter 3 of Sipser. Come up with (at least) two questions about the reading where you're not sure of the answer. These may relate to points where the book is confusing, or simply to some related question or conjecture that occurs to you while doing the reading.

2. Do Problem 3.15(b–e) on page 161 of Sipser. You don't need to do 3.15(a) because its answer is on page 163.

3. Problem 3.16(b–d) on page 161 of Sipser. You don't need to do 3.16(a) because its answer is on page 163.

4. Do Problem 3.18 (page 162) in Sipser.

5. Describe (clearly, but informally) how a finite state machine with *two* stacks could simulate a Turing Machine. (Big hint: consider the part of the tape to the left of the read/write head, and the part of the tape to the right.)

6. Goldbach's Conjecture states that every even integer greater than 2 can be expressed as the sum of two primes. It has not yet been proved, but (according to Wikipedia) all even numbers up to 1.609×10^{18} have been checked so far, and each has turned out to be a sum of primes. Explain how, given a (sadly, mythical) Halting checker, we could determine the truth of Goldbach's Conjecture once and for all.

7. **The Elusiveness of Undecidability.** It is widely known that Halting (membership in $H = \{ \langle M, w \rangle \mid M \text{ halts on input } w \}$) is not decidable. But:

(a) Consider the following machine, which we will call $\texttt{TM}_1$. It first confirms that the tape contains a number expressed in binary (and if not, rejects). It then enters the following loop:

- All digits are zero, it halts and accepts (ending the loop).
- Otherwise, it decrements the binary number by 1 (by completely overwriting it.)

Does the machine $\texttt{TM}_1$ halt on the input 10000000000000?

(b) Consider a similar machine, which we will call TM_2. It is exactly the same as TM_1, except that each time around the loop it *increments* the binary number by 1. Does the machine $\texttt{TM}_2$ halt on the input 10000000000000?

(c) Why don't your answers to Parts (a) and (b) contradict the undecidability of Halting?

(d) $\texttt{TM}_1$ and $\texttt{TM}_2$ are admittedly not very interesting. You are probably wondering, since Halting is undecidable, where the hard Turing Machines are. Suppose I were to design a very complicated $\texttt{TM}_3$, and prove mathematically that "there is no procedure by which you can determine whether $\texttt{TM}_3$ terminates on the input 10000000000000" (essentially, that "you will not and cannot know, even in principle, whether TM_3 terminates on this input"). Show that this supposition leads to a contradiction, and so I couldn't produce such a proof for any specific Turing Machine.