CS 6505: Computability & Algorithms
Lecture 2, January 13, 2010

Undecidability

In this lecture we will see that some well-defined functions cannot be computed. First we need a model of computation. We could use a Turing Machine, or not. (See Figure 1)

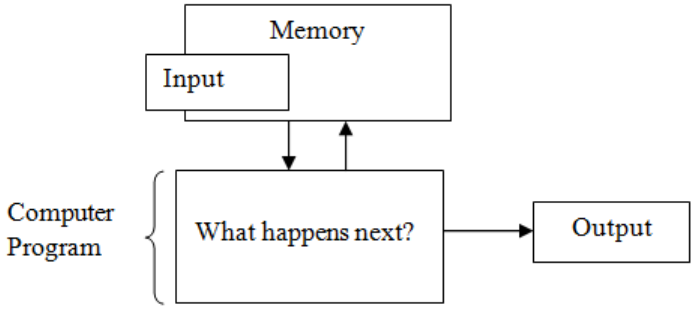

Figure 1: Program Model

We can model the program itself as a finite state machine as follows:

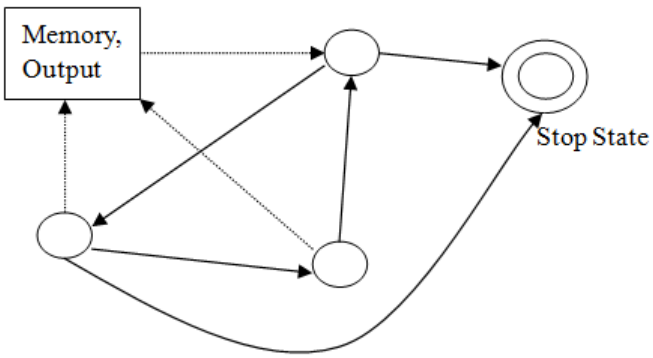

Figure 2: Program as Finite State Machine. Program states are represented as ellipses, and transitions between states are shown as solid lines. Interactions with memory and writes to output are shown as dashed lines. The top right state is the stop state, where the program will terminate.

The current full state is a vertex (state) of this directed graph plus the contents of the memory. (See Figure 2.) Together they determine the next state and changes to the memory (if any) and next output (if any). Memory, input, and output are all in a finite alphabet (e.g., {0,1}).

Universal Machine

Programs can be executed step by step. To run these programs, we use a *Universal Machine*. A Universal Machine U takes any program P and input I_p and then executes P on I_p.

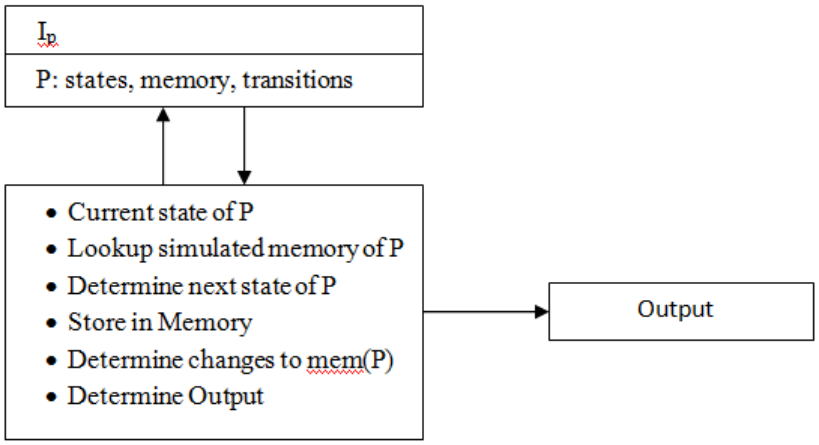

Figure 3: Universal Machine U runs program P on I_p.

U itself is also a program.
Now let ρ be the set of all programs P with inputs I_p. Now define

$$f : \rho \rightarrow \{0, 1\}$$

$$f(P, I_p) = \begin{cases} 1 & P \text{ halts at some point on} I_p \\ 0 & P \text{ runs forever on} I_p \end{cases}$$

Is there a program for f? Is f computable (in this model)?
For simplicity, consider programs with null inputs:

$$f(P) = \begin{cases} 1 & P \text{ halts at some point} \\ 0 & P \text{ runs forever} \end{cases}$$

Now suppose that a program H computes f. Define Z as follows:

- Run H on Z
- if $H(Z) = 1$ (i.e., H "halts") then run forever
- else halt

Notice that Z gives its own description to H in the first step of the algorithm. This may look odd, but we use this construction to obtain our contradiction.

Now, does Z halt?

By our definition, Z halts iff Z does not halt! This is a contradiction. Therefore such an H does not exist!

Thm 1 : The Halting problem is undecidable (in this model of computation).

Here's how we can describe the size and countability of the sets that we've seen in the last couple of days:

UNCOUNTABLE $\leftarrow$ Languages $\subseteq$ Functions

Programs $\subseteq$ Strings $\rightarrow$ COUNTABLE

Relevant Readings

- Sipser, Chapter 4
- Papadimitriou, Chapter 3