# CS/ENGRI 172, Fall 2002
# 10/4/02: Lecture Sixteen Handout

**Topics**: The impossibility of computing the halting function (cont).

## Adaptations of Kleinberg and Papadimitrious's "no Universal Termination Detector" argument

We slightly modify the argument from Kleinberg and Papadimitriou for pedagogical (and obscure technical) reasons. For completeness, we outline the differences here.

- Our list $M_1, M_2, \ldots$ is the (infinite) list of all Turing machine programs that take a sequence of A's as input, rather than of all Turing machine programs. (The ordering of the programs is still induced by their length, breaking ties alphabetically, as in the reading.)
- In general, where the reading talks about an input $n$ or $k$, we "translate" that to mean a sequence of $n$ or $k$ A's. Basically, we are just enforcing a particular encoding convention on the inputs to the machines (programs) on our list.
- We explicitly define the *halting function*:

$$h(M_i, j) = \begin{cases} 1 \text{ (yes)}, & \text{if } M_i \text{ would halt given } j \text{ A's as input} \\ 0 \text{ (no)} & \text{if } M_i \text{ would not halt given } j \text{ A's as input} \end{cases}$$

  So our theorem is about the fact that there cannot be any Turing machine (and hence, there cannot be a computer) that computes the halting function.

  Note again that the halting function itself definitely exists (as much as any mathematical function can be said to exist); the point of our theorem is that *there isn't any computational means to figure out what the halting function's value is for all of its possible inputs.*

  This is almost surely different than what one has seen before. In our early education, we usually associate a function with a means for computing it; for example, we learn that the "squaring" function is the result of multiplying a number by itself. But by being precise about a model for computation, computer scientists were able to show that the notion of function and the notion of computation are in fact distinct.