Computation, Information, and Intelligence (ENGRI/CS/INFO/COGST 172), Spring 2007
4/27/07: **Lecture 39 aid — Limits of computation**

Topics: the incomputability of the halting function; the upside of impossibility: zero-knowledge protocols
Announcements: Please complete a course evaluation for this class on-line at

http://www.engineering.cornell.edu/CourseEval/crseval/

and be assured not only of our own gratitude for the feedback, but also potentially the thanks of generations of future students. Infinitely more useful to us than the numerical responses are any concrete feedback comments you give to the text-response questions. The polls close sometime on Sunday.

I. Aside: An example of enumeration for a simple concept Here is an illustration of the output of a not particularly clever mechanical procedure for the enumeration of sequences of a's, b's, and c's in which the number of a's is bigger than the number of b's; as in the case of enumerating "B-input" TMs, the list puts shortest sequences first, with ties broken alphabetically.

$$\underbrace{a}_{s_1}, \cancel{b}, \cancel{c}, \underbrace{aa}_{s_2}, \cancel{ab}, \underbrace{ac}_{s_3}, \cancel{ba}, \cancel{bb}, \cancel{bc}, \underbrace{ca}_{s_4}, \cancel{cb}, \cancel{cc}, \underbrace{aaa}_{s_5}, \underbrace{aab}_{s_6}, \underbrace{aac}_{s_7}, \underbrace{aba}_{s_8}, \cancel{abb}, \cancel{abc}, \ldots$$

It should be easy to see how we could create a procedure that, given a number i, figures out what s_i is by going through the enumeration procedure implicitly described above: just mechanically produce *all* sequences of symbols one by one (shortest first), checking to see if each such symbol passes the requisite syntactic check. Similarly, we can create a procedure that, given i, returns the i^{th} B-input TM M_i.

II. Reminder: the halting function M_i denotes the i^{th} B-input TM.

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

Note that by "halting", we mean "would halt in a finite number of steps".

The idea behind our proof is to show that *if* the halting function were computable by some "termination detector" D, then an *impossible* TM would have to exist, namely ...

III. The evil machine X Suppose a termination detector D exists. Then we could construct a Turing machine X that uses D as a subcomponent and that, when given ℓ B's as input,

(a) recovers the program for M_ℓ;
(b) runs D to determine the value (0 or 1) of $h(M_\ell, \ell)$; and
(c1) if that value is "1", enters an artificial infinite loop
(c2) if that value is "0", sets its output to 1 and halts

IV. "Zero knowledge" protocol for 3-colorability

You declare your three colors (e.g., red, green, blue).
($\star$) Off-stage, randomly permute your coloring (e.g., red $\leftrightarrow$ green, blus stays the same).
Present color-hidden graph.
Suspicious entity chooses an edge.
You reveal the edge's two endpoints (they ought to be different colors, and from your declared set).