Computation, Information, and Intelligence (COMS/ENGRI/INFO/COGST 172), Fall 2005
11/21/05: **Lecture 36 aid — Turing machines, continued**

Agenda: Examples and definitions; computability of functions; from machines to programs.

I. Example one-tape Turing machine We specify the state set as "carry" and "no-carry"; the start state as "carry"; the allowable input symbols as 0, 1,...,9 (note that "blank" should not be an allowable input symbol); we'll ignore other details that would be required in a full specification; and specify the machine's behavior as follows.

> If reading a "0" and in state "carry", write "1", change to state "no-carry", stay put.
> If reading a "1" and in state "carry", write "2", change to state "no-carry", stay put.
> If reading a "2" and in state "carry", write "3", change to state "no-carry", stay put.
> ...
> If reading a "9" and in state "carry", write "0", stay in state "carry", move right.
> If reading a "blank" and in state "carry", write "1", change to state "no-carry", stay put.

(In a way, we are allowing a completely blank tape to represent the input number zero even though we have a symbol "0", but for simplicity we have elected not to deal with this issue.)

II. Another example one-tape TM For brevity, we'll skip most of the initial specification that should be given. We'll assume there's a special marker "!" at the beginning of the tape. The start state is "no-carry".

end-of-tape rule:

> If reading a "!" and in state "no-carry", write "!", change to state "carry", move right.

carry rules:

> If reading a "0" and in state "carry", write "1", change to state "no-carry", move left.
> If reading a "1" and in state "carry", write "2", change to state "no-carry", move left.
> If reading a "2" and in state "carry", write "3", change to state "no-carry", move left.
> ...
> If reading a "9" and in state "carry", write "0", stay in state "carry", move right.
> If reading a "blank" and in state "carry", write "1", change to state "no-carry", move left.

return-to-tape-end rules:

> If reading a "0" and in state "no-carry", write "0", stay in state "no-carry", move left.
> If reading a "1" and in state "no-carry", write "1", stay in state "no-carry", move left.
> If reading a "2" and in state "no-carry", write "2", stay in state "no-carry", move left.
> ...
> If reading a "9" and in state "no-carry", write "9", stay in state "no-carry", move left.

III. Definition of TM function computation Let $f : D \to R$ be a function. A Turing machine M *computes* f if, for every $x \in D$, when M is initialized with input x, it eventually *halts* — ends up in a situation where no rule applies — with $f(x)$ on its (output) tape.

We will also allow for encodings of x and $f(x)$. For example, we might have a Turing machine that, given n "hash marks" as input, returns n^2 "hash marks"; we would then still say that the Turing machine computes $f(x) = x^2$, where x is a non-negative integer.

IV. Enumeration of "A-machines" We denote by $M_1, M_2, \ldots$ a infinite list (with no repeats) of *all* TMs that take only sequences of A's as input and produce sequences of A's as output. The list is such that given a (suitably encoded) number i, it is possible to recover the program for M_i (i.e., there exists a Turing machine that does the job).

The sequences of A's are intended to be encodings of non-negative integers. Thus, *all computable functions from the non-negative integers to the non-negative integers are represented in the list.*

V. The halting function

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