CS/ENGRI 172, Fall 2003: Computation, Information, and Intelligence
10/1/03: Turing Machines

Example Turing Machine Definition

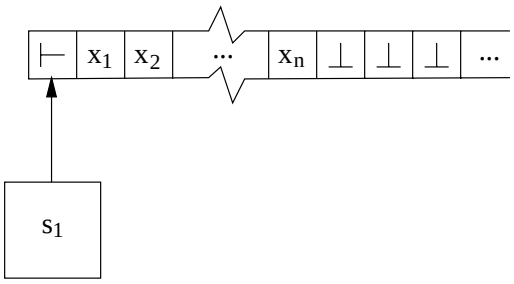
This Turing machine has two internal states, q_1 and q_2, with q_1 being the initial state. The Turing machine's symbols are A, B, $\vdash$, and $\perp$. The control table for the Turing machine is:

	$\vdash$	A	B	$\perp$
q_1	(q_1, $\vdash$, R)	(q_2, B, R)	(q_1, B, R)	(q_1, $\perp$, R)
q_2	—	(q_1, A, R)	(q_2, B, R)	(q_2, $\perp$, R)

Note that this Turing machine never halts.

Turing Machine Initialization

Suppose we have a Turing machine M with m distinct states $s_1, s_2, \ldots, s_m$, where s_1 is the initial state, and that M has a symbol alphabet consisting of l distinct symbols $a_1, a_2, \ldots, a_l$, with a_1 being the left-end marker $\vdash$ and $a_2 = \perp$ (we assume $l \geq 3$ and $m \geq 1$). Then a legal input to M could be $x = x_1 x_2 \ldots x_n$, with each x_i drawn from among $a_3, \ldots, a_l$ (so the input can't contain blanks or the left-end marker, but repeats are allowed), and the *initial configuration* of M on this input would look like:



Definition: A Turing machine *computes* a function $f : D \rightarrow R$ (where D is the domain of the function and R is the range) if, given any element d in D as input, it *halts* with $f(d)$ left-justified on the tape, proceeded by $\vdash$ and followed only by $\perp$.

Computer scientists believe that, for any reasonable (finite-precision) function, there is some Turing machine that computes it.

Example Turing Machine Algorithm

When designing a Turing machine, it is helpful to break the problem down into discrete pieces, each of which we can map to separate states in the Turing machine. Here is an example of an algorithm for doubling the number of A's in an input:

1. scan right to find the first blank
2. search left for the first A
3. mark the A and copy it to the first blank (we'll then repeat step 2)
4. (if step 2 didn't find an A, then A's are exhausted) clean up