mac.1 Turing Machines

tur:mac:tur:sec

The formal definition of what constitutes a Turing machine looks abstract, but is actually simple: it merely packs into one mathematical structure all the information needed to specify the workings of a Turing machine. This includes (1) which states the machine can be in, (2) which symbols are allowed to be on the tape, (3) which state the machine should start in, and (4) what the instruction set of the machine is. explanation

Definition mac.1 (Turing machine). A *Turing machine* M is a tuple $\langle Q, \Sigma, q_0, \delta\rangle$ consisting of

1. a finite set of *states* Q,
2. a finite *alphabet* Σ which includes $\triangleright$ and 0,
3. an *initial state* $q_0 \in Q$,
4. a finite *instruction set* $\delta\colon Q \times \Sigma \nrightarrow Q \times \Sigma \times \{L, R, N\}$.

The partial function δ is also called the *transition function* of M.

We assume that the tape is infinite in one direction only. For this reason it is useful to designate a special symbol $\triangleright$ as a marker for the left end of the tape. This makes it easier for Turing machine programs to tell when they're "in danger" of running off the tape. We could assume that this symbol is never overwritten, i.e., that $\delta(q, \triangleright) = \langle q', \triangleright, x\rangle$ if $\delta(q, \triangleright)$ is defined. Some textbooks do this, we do not. You can simply be careful when constructing your Turing machine that it never overwrites $\triangleright$. Moreover, there are cases where allowing such overwriting provides some convenient flexibility. explanation

Example mac.2. *Even Machine:* The even machine is formally the quadruple $\langle Q, \Sigma, q_0, \delta\rangle$ where

$$\begin{aligned} Q &= \{q_0, q_1\} \\ \Sigma &= \{\triangleright, 0, 1\}, \\ \delta(q_0, 1) &= \langle q_1, 1, R\rangle, \\ \delta(q_1, 1) &= \langle q_0, 1, R\rangle, \\ \delta(q_1, 0) &= \langle q_1, 0, R\rangle. \end{aligned}$$

Photo Credits

Bibliography