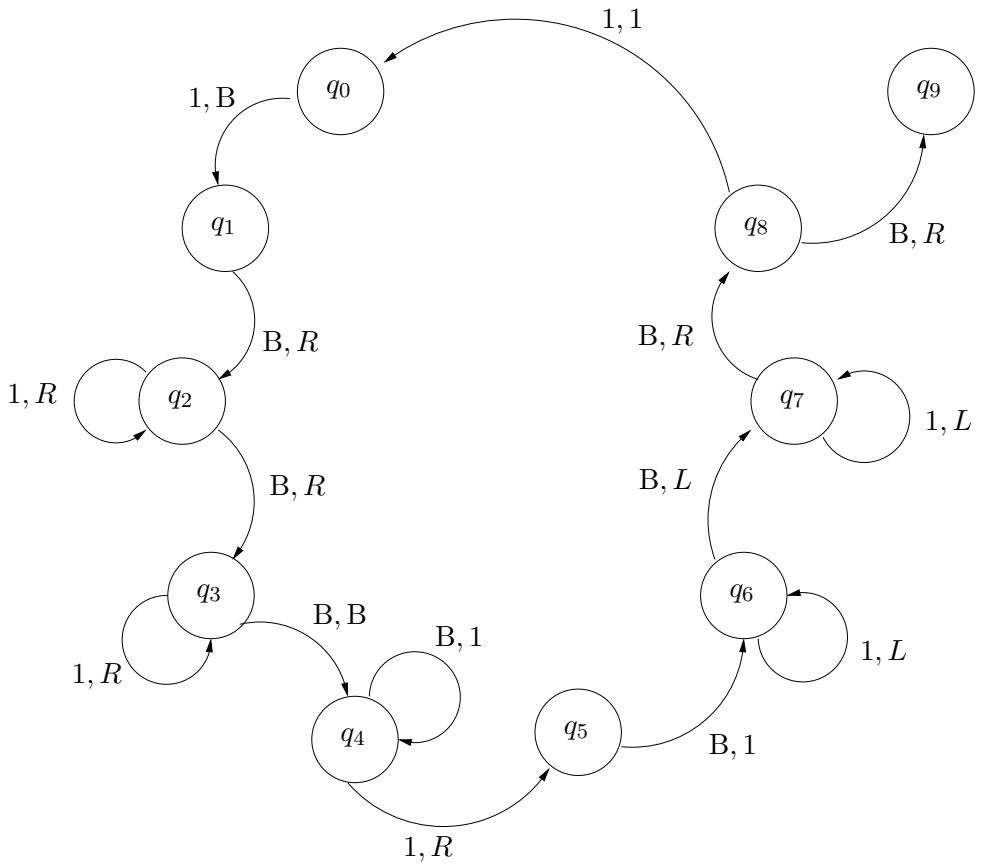
Turing Machines as state transition diagrams

Dirk Schlimm

November 5, 2017

Turing Machines can be represented by a list of quadrupels, or as state transition diagrams. For example, the following diagram represents a TM, with alphabet $\{\mathrm{B}, 1\}$.



The nodes represent the states (q_0, q_1, q_2, etc.) of the TM. An arrow from state q_i to q_j labeled 'r, p' means: If the TM is in state q_i and reads the symbol $r \in \{\mathrm{B}, 1\}$, then it executes the operation $p \in \{\mathrm{B}, 1, R, L\}$ and changes into state q_j. Thus, each arrow in the diagram corresponds to a single TM instruction. For example, in the diagram, the arrow labeled '$1, \mathrm{B}$' from q_0 to q_1 corresponds to the TM instruction

$$(\ q_0,\ 1,\ \mathrm{B},\ q_1\ ).$$

The next arrows correspond to the instructions (q_1, B, R, q_2), (q_2, 1, R, q_2), etc.

What does this TM do?

The Halting Problem

Dirk Schlimm

November 5, 2017

Given an enumeration T_1, T_2, T_3, ... of Turing Machines (TM), we define the following set K of natural numbers (where $T_2(5)$ means that the second TM runs on input 5, etc.):

$$K \Leftrightarrow_{df} \{x \in \mathbb{N} \mid T_x(x) \downarrow\}.$$

This is the set of indices of TMs that eventually *halt* ('$\downarrow$') if the input is their own index in the enumeration. This set is called the *halting set*. (See *Hamilton*, p. 181.)

Big question (the Halting Problem): Is there a TM that can *decide* for every number $x \in \mathbb{N}$, whether $x \in K$ or $x \notin K$? By 'decide' is meant that the TM gives a definite answer (e. g., 1 for 'yes' and 0 for 'no') for each input x after a finite amount of time; this TM does not loop (i. e., is undefined, '$\uparrow$') on any input; instead, it is everywhere defined, i. e., *total*.

Dovetailing. Note, that the set K can be *enumerated* by a procedure called *dovetailing*: We go through the table below as we did in the proof showing that the set of rational numbers is countable and we run each TM for 1, 2, 3, etc. steps. Whenever we find that one of the TMs in the table halts after a finite number of steps, we output its index. Since every TM that halts, halts after a finite number of executed instructions, it is eventually found in this way. So, all elements of K can be enumerated (or listed) by this procedure.

	Number of executed instructions			
	1	2	3	⋯
T_1 with input 1	$T_1(1)$	$T_1(1)$	$T_1(1)$	⋯
T_2 with input 2	$T_2(2)$	$T_2(2)$	$T_2(2)$	⋯
⋮	$T_3(3)$	$T_3(3)$	⋯	

Claim: The set K of natural numbers cannot be decided by a TM.

Proof by contradiction: Assume (for the sake of contradiction) that there is a TM, say T_H, that decides for each $x \in \mathbb{N}$ whether $x \in K$ or $x \notin K$. Then,

$$T_H(x) = \begin{cases} 1, & \text{if } x \in K, \text{ i. e., } T_x(x) \text{ halts,} \qquad \text{i. e., } T_x(x) \downarrow, \\ 0, & \text{if } x \notin K, \text{ i. e., } T_x(x) \text{ does not halt, i. e., } T_x(x) \uparrow. \end{cases}$$

Using T_H we can easily build a new TM, say T_G, that behaves as follows:

$$T_G(x) = \begin{cases} 0, & \text{if } T_H(x) = 0, \\ \uparrow, & \text{if } T_H(x) = 1. \end{cases}$$

This TM must occur in our original enumeration of TMs, say at position n. So,

$$T_G(x) = T_n(x).$$

Now, what is the output of TM T_G on input n?

Due to the way we have defined T_G, there are only two possibilities:

a) $T_G(n) = 0$, i. e., $T_n(n) \downarrow$, if $T_H(n) = 0$, i. e., $T_n(n) \uparrow$. $\lightning$
b) $T_G(n) = \uparrow$, i. e., $T_n(n) \uparrow$, if $T_H(n) = 1$, i. e., $T_n(n) \downarrow$. $\lightning$

Since both cases lead to a contradiction, we conclude that our initial assumption was false.

Conclusion: There is no TM that decides for every $x \in \mathbb{N}$, whether $x \in K$ or $x \notin K$. $\square$

This problem is *not computable by a Turing Machine*, and by Turing's Thesis, *uncomputable*.