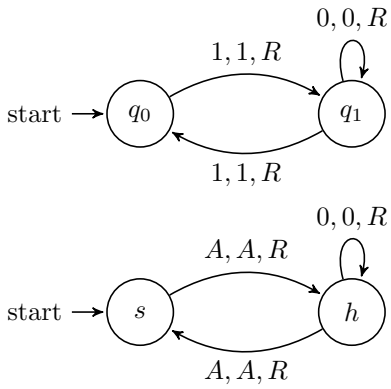

Figure 1: Variants of the *Even* machine

tur:und:enu:fig:variants

und.1 Enumerating Turing Machines

tur:und:enu:sec

explanation

We can show that the set of all Turing machines is enumerable. This follows from the fact that each Turing machine can be finitely described. The set of states and the tape vocabulary are finite sets. The transition function is a partial function from $Q \times \Sigma$ to $Q \times \Sigma \times \{L, R, N\}$, and so likewise can be specified by listing its values for the finitely many argument pairs for which it is defined.

This is true as far as it goes, but there is a subtle difference. The definition of Turing machines made no restriction on what elements the set of states and tape alphabet can have. So, e.g., for every real number, there technically is a Turing machine that uses that number as a state. However, the *behavior* of the Turing machine is independent of which objects serve as states and vocabulary. Consider the two Turing machines in Figure 1. These two diagrams correspond to two machines, M with the tape alphabet $\Sigma = \{\triangleright, 0, 1\}$ and set of states $\{q_0, q_1\}$, and M' with alphabet $\Sigma' = \{\triangleright, 0, A\}$ and states $\{s, h\}$. But their instructions are otherwise the same: M will halt on a sequence of n 1's iff n is even, and M' will halt on a sequence of n A's iff n is even. All we've done is rename 1 to A, q_0 to s, and q_1 to h. This example generalizes: we can think of Turing machines as the same as long as one results from the other by such a renaming of symbols and states. In fact, we can simply think of the symbols and states of a Turing machine as positive integers: instead of σ_0 think 1, instead of σ_1 think 2, etc.; $\triangleright$ is 1, 0 is 2, etc. In this way, the *Even* machine becomes the machine depicted in Figure 2. We might call a Turing machine with states and symbols that are positive integers a *standard* machine, and only consider standard machines from now on.[1]

We wanted to show that the set of Turing machines is enumerable, and

[1] The terminology "standard machine" is not standard.

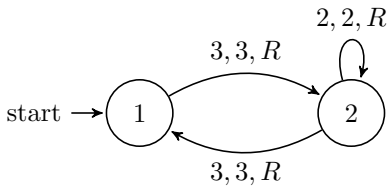

Figure 2: A standard *Even* machine

tur:und:enu:
fig:standard-even

with the above considerations in mind, it is enough to show that the set of standard Turing machines is enumerable. Suppose we are given a standard Turing machine $M = \langle Q, \Sigma, q_0, \delta \rangle$. How could we describe it using a finite string of positive integers? We'll first list the number of states, the states themselves, the number of symbols, the symbols themselves, and the starting state. (Remember, all of these are positive integers, since M is a standard machine.) What about δ? The set of possible arguments, i.e., pairs $\langle q, \sigma \rangle$, is finite, since Q and Σ are finite. So the information in δ is simply the finite list of all 5-tuples $\langle q, \sigma, q', \sigma', d \rangle$ where $\delta(q, \sigma) = \langle q', \sigma', D \rangle$, and d is a number that codes the direction D (say, 1 for L, 2 for R, and 3 for N).

In this way, every standard Turing machine can be described by a finite list of positive integers, i.e., as a sequence $s_M \in (\mathbb{Z}^+)^*$. For instance, the standard *Even* machine is coded by the sequence

$$2, \underbrace{1, 2}_{Q}, 3, \overbrace{1, 2, 3}^{\Sigma}, 1, \underbrace{1, 3, 2, 3, 2}_{\delta(1,3)=\langle 2,3,R\rangle}, \overbrace{2, 2, 2, 2, 2}^{\delta(2,2)=\langle 2,2,R\rangle}, \underbrace{2, 3, 1, 3, 2}_{\delta(2,3)=\langle 1,3,R\rangle}.$$

Theorem und.1. *There are functions from* $\mathbb{N}$ *to* $\mathbb{N}$ *which are not Turing computable.*

Proof. We know that the set of finite sequences of positive integers $(\mathbb{Z}^+)^*$ is enumerable (**??**). This gives us that the set of descriptions of standard Turing machines, as a subset of $(\mathbb{Z}^+)^*$, is itself enumerable. Every Turing computable function $\mathbb{N}$ to $\mathbb{N}$ is computed by some (in fact, many) Turing machines. By renaming its states and symbols to positive integers (in particular, $\triangleright$ as 1, 0 as 2, and 1 as 3) we can see that every Turing computable function is computed by a standard Turing machine. This means that the set of all Turing computable functions from $\mathbb{N}$ to $\mathbb{N}$ is also enumerable.

On the other hand, the set of all functions from $\mathbb{N}$ to $\mathbb{N}$ is not enumerable (**??**). If all functions were computable by some Turing machine, we could enumerate the set of all functions by listing all the descriptions of Turing machines that compute them. So there are some functions that are not Turing computable. □

Problem und.1. Can you think of a way to describe Turing machines that does not require that the states and alphabet symbols are explicitly listed? You may define your own notion of "standard" machine, but say something about why every Turing machine can be computed by a "standard" machine in your new sense.

Photo Credits

Bibliography