und.1 The Halting Problem

tms:und:hal: sec

Assume we have fixed some finite descriptions of Turing machines. Using these, we can enumerate Turing machines via their descriptions, say, ordered by the lexicographic ordering. Each Turing machine thus receives an *index*: its place in the enumeration M_1, M_2, M_3, ... of Turing machine descriptions. explanation

We know that there must be non-Turing-computable functions: the set of Turing machine descriptions—and hence the set of Turing machines—is enumerable, but the set of all functions from $\mathbb{N}$ to $\mathbb{N}$ is not. But we can find specific examples of non-computable function as well. One such function is the halting function.

Definition und.1 (Halting function). The *halting function* h is defined as

$$h(e,n) = \begin{cases} 0 & \text{if machine } M_e \text{ does not halt for input } n \\ 1 & \text{if machine } M_e \text{ halts for input } n \end{cases}$$

Definition und.2 (Halting problem). The *Halting Problem* is the problem of determining (for any e, n) whether the Turing machine M_e halts for an input of n strokes.

We show that h is not Turing-computable by showing that a related function, s, is not Turing-computable. This proof relies on the fact that anything that can be computed by a Turing machine can be computed using just two symbols: 0 and 1, and the fact that two Turing machines can be hooked together to create a single machine. explanation

Definition und.3. The function s is defined as

$$s(e) = \begin{cases} 0 & \text{if machine } M_e \text{ does not halt for input } e \\ 1 & \text{if machine } M_e \text{ halts for input } e \end{cases}$$

Lemma und.4. *The function s is not Turing computable.*

Proof. We suppose, for contradiction, that the function s is Turing-computable. Then there would be a Turing machine S that computes s. We may assume, without loss of generality, that when S halts, it does so while scanning the first square. This machine can be "hooked up" to another machine J, which halts if it is started on a blank tape (i.e., if it reads 0 in the initial state while scanning the square to the right of the end-of-tape symbol), and otherwise wanders off to the right, never halting. $S \frown J$, the machine created by hooking S to J, is a Turing machine, so it is M_e for some e (i.e., it appears somewhere in the enumeration). Start M_e on an input of e 1s. There are two possibilities: either M_e halts or it does not halt.

1. Suppose M_e halts for an input of e 1s. Then $s(e) = 1$. So S, when started on e, halts with a single 1 as output on the tape. Then J starts with a 1

on the tape. In that case J does not halt. But M_e is the machine $S \frown J$, so it should do exactly what S followed by J would do. So M_e cannot halt for an input of e 1's.

2. Now suppose M_e does not halt for an input of e 1s. Then $s(e) = 0$, and S, when started on input e, halts with a blank tape. J, when started on a blank tape, immediately halts. Again, M_e does what S followed by J would do, so M_e must halt for an input of e 1's.

This shows there cannot be a Turing machine S: s is not Turing computable.□

Theorem und.5 (Unsolvability of the Halting Problem). *The halting problem is unsolvable, i.e., the function h is not Turing computable.* *tms:und:hal:thm:halting-problem*

Proof. Suppose h were Turing computable, say, by a Turing machine H. We could use H to build a Turing machine that computes s: First, make a copy of the input (separated by a blank). Then move back to the beginning, and run H. We can clearly make a machine that does the former, and if H existed, we would be able to "hook it up" to such a modified doubling machine to get a new machine which would determine if M_e halts on input e, i.e., computes s. But we've already shown that no such machine can exist. Hence, h is also not Turing computable. □

Problem und.1. The Three Halting (3-Halt) problem is the problem of giving a decision procedure to determine whether or not an arbitrarily chosen Turing Machine halts for an input of three strokes on an otherwise blank tape. Prove that the 3-Halt problem is unsolvable.

Problem und.2. Show that if the halting problem is solvable for Turing machine and input pairs M_e and n where $e \neq n$, then it is also solvable for the cases where $e = n$.

Problem und.3. We proved that the halting problem is unsolvable if the input is a number e, which identifies a Turing machine M_e via an enumaration of all Turing machines. What if we allow the description of Turing machines from **??** directly as input? (This would require a larger alphabet of course.) Can there be a Turing machine which decides the halting problem but takes as input descriptions of Turing machines rather than indices? Explain why or why not.

Photo Credits

Bibliography