

thy.1 The Halting Problem

cmp:thy:halt:
sec

Since, in our construction, $\text{Un}(k, x)$ is defined if and only if the computation of the function coded by k produces a value for input x , it is natural to ask if we can decide whether this is the case. And in fact, it is not. For the Turing machine model of computation, this means that whether a given Turing machine halts on a given input is computationally undecidable. The following theorem is therefore known as the “undecidability of the halting problem.” I will provide two proofs below. The first continues the thread of our previous discussion, while the second is more direct.

cmp:thy:halt:
thm:halting-problem

Theorem thy.1. *Let*

$$h(k, x) = \begin{cases} 1 & \text{if } \text{Un}(k, x) \text{ is defined} \\ 0 & \text{otherwise.} \end{cases}$$

Then h is not computable.

Proof. If h were computable, we would have a universal computable function, as follows. Suppose h is computable, and define

$$\text{Un}'(k, x) = \begin{cases} fnUn(k, x) & \text{if } h(k, x) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

But now $\text{Un}'(k, x)$ is a total function, and is computable if h is. For instance, we could define g using primitive recursion, by

$$\begin{aligned} g(0, k, x) &\simeq 0 \\ g(y + 1, k, x) &\simeq \text{Un}(k, x); \end{aligned}$$

then

$$\text{Un}'(k, x) \simeq g(h(k, x), k, x).$$

And since $\text{Un}'(k, x)$ agrees with $\text{Un}(k, x)$ wherever the latter is defined, Un' is universal for those partial computable functions that happen to be total. But this contradicts ?? □

Proof. Suppose $h(k, x)$ were computable. Define the function g by

$$g(x) = \begin{cases} 0 & \text{if } h(x, x) = 0 \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The function g is partial computable; for example, one can define it as $\mu y h(x, x) = 0$. So, for some k , $g(x) \simeq \text{Un}(k, x)$ for every x . Is g defined at k ? If it is, then, by the definition of g , $h(k, k) = 0$. By the definition of f , this means that $\text{Un}(k, k)$ is undefined; but by our assumption that $g(k) \simeq \text{Un}(k, x)$ for every x , this means that $g(k)$ is undefined, a contradiction. On the other hand, if $g(k)$ is undefined, then $h(k, k) \neq 0$, and so $h(k, k) = 1$. But this means that $\text{Un}(k, k)$ is defined, i.e., that $g(k)$ is defined. □

explanation We can describe this argument in terms of Turing machines. Suppose there were a Turing machine H that took as input a description of a Turing machine K and an input x , and decided whether or not K halts on input x . Then we could build another Turing machine G which takes a single input x , calls H to decide if machine x halts on input x , and does the opposite. In other words, if H reports that x halts on input x , G goes into an infinite loop, and if H reports that x doesn't halt on input x , then G just halts. Does G halt on input G ? The argument above shows that it does if and only if it doesn't—a contradiction. So our supposition that there is a such Turing machine H , is false.

Photo Credits

Bibliography