

thy.1 Totality is Undecidable

cmp:thy:tot:
sec Let us consider one more example of using the *s-m-n* theorem to show that something is noncomputable. Let Tot be the set of indices of total computable functions, i.e.

$$\text{Tot} = \{x : \text{for every } y, \varphi_x(y) \downarrow\}.$$

cmp:thy:tot:
prop:total **Proposition thy.1.** *Tot is not computable.*

Proof. To see that Tot is not computable, it suffices to show that *K* is reducible to it. Let $h(x, y)$ be defined by

$$h(x, y) \simeq \begin{cases} 0 & \text{if } x \in K \\ \text{undefined} & \text{otherwise} \end{cases}$$

Note that $h(x, y)$ does not depend on y at all. It should not be hard to see that h is partial computable: on input x, y , we compute h by first simulating the function φ_x on input x ; if this computation halts, $h(x, y)$ outputs 0 and halts. So $h(x, y)$ is just $Z(\mu s T(x, x, s))$, where Z is the constant zero function.

Using the *s-m-n* theorem, there is a primitive recursive function $k(x)$ such that for every x and y ,

$$\varphi_{k(x)}(y) = \begin{cases} 0 & \text{if } x \in K \\ \text{undefined} & \text{otherwise} \end{cases}$$

So $\varphi_{k(x)}$ is total if $x \in K$, and undefined otherwise. Thus, k is a reduction of K to Tot. \square

It turns out that Tot is not even computably enumerable—its complexity lies further up on the “arithmetic hierarchy.” But we will not worry about this strengthening here. digression

Photo Credits

Bibliography