

-NoValue-

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, O.s.,

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

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

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

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

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 $\text{zero}(\mu s \, T(x, x, s))$, where zero 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 ,

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

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 digression lies further up on the “arithmetical hierarchy.” But we will not worry about this strengthening here.

Créditos das Imagens

Referências Bibliográficas