und.1 Introdução

tur:und:int:sec

Pode parecer óbvio que nem toda função, nem mesmo toda função aritmética, pode ser computável. Há simplesmente um número grande demais delas, cujo comportamento é complicado demais. Funções definidas a partir do decaimento de partículas radioativas, por exemplo, ou de outro comportamento caótico ou aleatório. Suponha que comecemos a contar intervalos de 1 segundo a partir de um dado instante e definamos a função $f(n)$ como o número de partículas no universo que decaem no n-ésimo intervalo de 1 segundo após aquele momento inicial. Esta parece ser uma candidata a função que jamais poderemos ter esperança de computar.

Mas uma coisa é não conseguir imaginar como se computaria tais funções, e outra bem diferente é de fato provar que elas são incomputáveis. Na verdade, mesmo funções que parecem irremediavelmente complicadas podem, em um sentido abstrato, ser computáveis. Por exemplo, suponha que o universo seja finito no tempo—algum dia, num futuro muito distante, o universo se contrairá em um único ponto, como preveem algumas teorias cosmológicas. Então há apenas um número finito (mas incrivelmente grande) de segundos a partir daquele momento inicial para os quais $f(n)$ está definida. E qualquer função que esteja definida para apenas finitas entradas é computável: poderíamos listar as saídas em uma grande tabela, ou codificá-la em um diagrama de transição de estados de máquina de Turing muito grande.

Frequentemente estamos interessados em casos especiais de funções cujos valores fornecem as respostas a perguntas de sim/não. Por exemplo, a pergunta "n é um número primo?" está associada à função

$$\text{primo}(n) = \begin{cases} 1 & \text{se } n \text{ é primo} \\ 0 & \text{caso contrário.} \end{cases}$$

Dizemos que uma pergunta de sim/não pode ser *efetivamente decidida* se a função associada de valores 1/0 é efetivamente computável.

Para provar matematicamente que há funções que não podem ser efetivamente computadas, ou problemas que não podem ser efetivamente decididos, é essencial fixar um modelo específico de computação e mostrar que há funções que ele não pode computar ou problemas que ele não pode decidir. Podemos mostrar, por exemplo, que nem toda função pode ser computada por máquinas de Turing, e nem todo problema pode ser decidido por máquinas de Turing. Podemos então apelar para a tese de Church-Turing para concluir que não apenas as máquinas de Turing não são poderosas o suficiente para computar toda função, como também nenhum procedimento efetivo o é.

A chave para provar tais resultados negativos é o fato de que podemos atribuir números às próprias máquinas de Turing. A maneira mais fácil de fazer isso é enumerá-las, talvez fixando um modo específico de escrever máquinas de Turing e seus programas e, em seguida, listando-os de maneira sistemática. Uma vez que vejamos que isso pode ser feito, então a existência de funções Turing-incomputáveis decorre de simples considerações de cardinalidade: o

conjunto de funções de $\mathbb{N}$ a $\mathbb{N}$ (na verdade, mesmo apenas de $\mathbb{N}$ a $\{0,1\}$) é não enumerável, mas como podemos enumerar todas as máquinas de Turing, o conjunto de funções Turing-computáveis é apenas infinito enumerável.

Também podemos definir funções e problemas *específicos* que podemos provar ser incomputáveis e indecidíveis, respectivamente. Um desses problemas é o assim chamado *Problema da Parada.* Máquinas de Turing podem ser finitamente descritas listando-se suas instruções. Tal descrição de uma máquina de Turing, isto é, um programa de máquina de Turing, pode é claro ser usada como entrada para outra máquina de Turing. Assim, podemos considerar máquinas de Turing que decidem perguntas sobre outras máquinas de Turing. Uma pergunta particularmente interessante é esta: "A máquina de Turing dada eventualmente para quando iniciada na entrada n?" Seria bom se houvesse uma máquina de Turing que pudesse decidir essa pergunta: pense nela como uma máquina de Turing de controle de qualidade que garante que as máquinas de Turing não fiquem presas em laços infinitos e coisas do tipo. O fato interessante, que Turing provou, é que não pode haver tal máquina de Turing. Não pode haver uma única máquina de Turing que, quando iniciada em uma entrada que consiste em uma descrição de uma máquina de Turing M e algum número n, sempre pare com saída 1 ou 0 conforme a máquina M teria parado ou não quando iniciada na entrada n.

Uma vez que tenhamos exemplos de problemas indecidíveis específicos, podemos usá-los para mostrar que outros problemas também são indecidíveis. Por exemplo, um célebre problema indecidível é a pergunta "A fórmula φ de primeira ordem é válida?". Não há máquina de Turing que, dada como entrada uma fórmula φ de primeira ordem, tenha garantia de parar com saída 1 ou 0 conforme φ seja válida ou não. Historicamente, a questão de encontrar um procedimento para resolver efetivamente esse problema foi chamada simplesmente de "o" problema da decisão; e assim dizemos que o problema da decisão é insolúvel. Turing e Church provaram esse resultado independentemente mais ou menos na mesma época, de modo que ele também é chamado de Teorema de Church-Turing.

Créditos das Imagens

Referências Bibliográficas