O Problema da Terminação de Programas

Paulo Mateus
Instituto Superior Técnico
e-mail: pmat@math.ist.utl.pt

Resumo: Neste artigo apresenta-se de forma simples e elementar a indecidibilidade do Problema da Terminação de Programas apresentado por Turing em 1936 em resposta ao Entscheidungsproblem de Hilbert. O resultado é apresentado no contexto de programas imperativos modernos em arquitectura de von Neumann, em vez de máquinas de Turing, para que o leitor que conhece rudimentos de programação se sinta confortável. Discute-se o impacto deste problema no desenvolvimento de aplicações computacionais. Termina-se o artigo utilizando o resultado para construir um número real não computável.

Abstract Herein we present and discuss the undecidability of the Halting Problem presented by Turing in 1935 to address Hilbert's Entscheidungsproblem. The result is presented in the context of imperative programs over von Neumann computer architecture instead of Turing machines. In this way we expect a reader with basic knowledge in programming to follow the details of the proof. The impacts of the result are discussed, namely on Hilbert's program and in the development of software applications. We conclude the paper by using the Halting problem to define an incomputable real number.

palavras-chave: programa; função computável; problema indecidível; real computável.

keywords: program, computable function, indecidible problem, computable real number.

1 Contexto e conceitos básicos

Quando em 1928 Hilbert e Ackermann apresentaram o famoso Entscheidungsproblem (problema da decisão), no seguimento do programa de Hilbert de formalizar a Matemática, estavam esperançados que todos problemas matemáticos pudessem ser resolvidos *mecanicamente*. Este objectivo remonta a Liebniz que, no século XVII, após ter construído com sucesso uma máquina de calcular mecânica ambicionava fazer uma máquina que determinasse a validade de fórmulas matemáticas. De forma sucinta, o Entscheidungsproblem consiste em determinar se existe um *procedimento mecânico*, que dadas

uma fórmula e uma teoria de primeira ordem retorna 1 se esta fórmula é válida na teoria e 0 caso contrário.

Em 1931, Gödel deu o primeiro passo para desmoronar as ambições de Hilbert ao mostrar que uma teoria de primeira ordem suficientemente rica não poderia ser finitamente axiomatizável [5]. Foi no entanto necessário esclarecer o que é um *procedimento mecânico*, ou mais rigorosamente, o que é um modelo de computação, um programa e uma função computável para responder de forma negativa ao Entscheidungsproblem. Em 1936, e de forma independente, Church [2] e Turing [10] propuseram dois modelos de computação e, capitalizando no resultado de Gödel referido atrás, mostraram que existem fórmulas na teoria da aritmética de Peano cuja validade não pode ser resolvida de *forma mecânica*. Mais tarde, outros modelos de computação foram propostos por Markov, Post, entre outros, mas tornou-se evidente que todos estes modelos eram equivalentes desde que se assumam alguns requisitos razoáveis, tais como, que um computador só pode manipular e armazenar dados com precisão finita. Esta equivalência é denominada por Postulado de Church-Markov-Turing e de forma resumida postula que todos os modelos de computação razoáveis são equivalentes, habitualmente designados por Turing equivalentes.

Antes de prosseguir é conveniente apresentar de forma mais rigorosa o que é um computador, um programa, uma função computável e um problema decidível. Para tal recorre-se ao modelo de computação no qual se baseou a construção do computador moderno, da autoria de von Neumann [6], chamado máquina R(andom) A(ccess) M(achine). O afastamento do modelo proposto por Turing, denominado por máquina de Turing, é propositado para que o leitor reconheça e entenda facilmente os conceitos. Para uma desmonstração ainda mais elegante, sem utilização de programas o leitor é aconselhado a ler [4].

Por computador entende-se uma máquina cuja memória é uma sucessão de registos nos quais se podem guardar números naturais. A configuração de memória é uma sucessão de números naturais

$$R_i,\ i \in \mathbb{N}$$

em que R_i é o valor que guardado no i-ésimo registo.

A configuração de memória pode ser alterada por intermédio da execução de programas. Os programas definem-se indutivamente da seguinte forma:

- Atribuições:
  - $R_i := 0$ (colocar a zero)

  - $R_i := R_i + 1$ (atribuir o sucessor)
  - $R_i := R_j$ (copiar registo)

- Composição sequencial de programas:

$$C_1; C_2$$

  onde C_1 e C_2 são programas.

- Composição alternativa de programas:

$$\texttt{if } (R_i = R_j) \texttt{ then } C_1 \texttt{ else } C_2$$

  onde C_1 e C_2 são programas.

- Composição iterativa de programas:

$$\texttt{while } (R_i \neq R_j) \texttt{ do } C$$

  onde C é um programa.

Denota-se o conjunto de todos o programas por $\mathcal{C}$.

A semântica intuitiva da execução de programas é mais ou menos óbvia para quem tem conhecimentos rudimentares de programação. As atribuições são programas atómicos que, ao serem executados, alteram a configuração de memória da seguinte forma: $R_i := 0$ coloca o i-ésimo registo a zero; $R_i := R_i + 1$ incrementa o i-ésimo registo por uma unidade; e $R_i := R_j$ copia o valor de registo R_j para R_i.

As composições controlam a ordem da execução das componentes do programa: $C_1; C_2$ executa o programa C_1 e depois o programa C_2; $\texttt{if } (R_i = R_j) \texttt{ then } C_1 \texttt{ else } C_2$ executa C_1 se o registo R_i tiver o mesmo valor que o registo R_j e C_2 caso contrário; $\texttt{while } (R_i \neq R_j) \texttt{ do } C$ executa C enquanto os registos R_i e R_j não têm o mesmo valor.

Note que a execução de um programa pode nunca terminar, em particular a execução do seguinte programa nunca termina, esteja a memória em que configuração estiver

$R_1 := R_2$;
$R_1 := R_1 + 1$;
$\texttt{while } (R_1 \neq R_2) \texttt{ do } R_1 := R_1$.

Um programa C computa uma função parcial $f_C^k : \mathbb{N}^k \to \mathbb{N}$ para cada $k \in \mathbb{N}$. O valor de $f_C^k(x_1, \ldots, x_k)$ com $(x_1, \ldots, x_n) \in \mathbb{N}^k$ é computado da seguinte forma:

- Primeiro, os registos de memória são inicializados de forma a que:
  - $R_i = x_i$ para $i = 1, \ldots, k$;
  - $R_i = 0$ para $i > k$.
- De seguida, o programa C é executado.
- Se o programa C não terminar a função está indefinida para o argumento $(x_1, \ldots, x_n)$.
- Se o programa C terminar o valor de $f_C^k(x_1, \ldots, x_k)$ é o valor que está guardado em R_1.

Um conjunto $A \subseteq \mathbb{N}^k$ é dito decidível se a sua função característica é computável, isto é, se a função

$$\chi_A(x) = \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{caso contrário} \end{cases}$$

é computável. É muito comum utilizar-se o termo *problema* para identificar um conjunto. Por exemplo, o problema da paridade corresponde ao problema de determinar se um número é par, ou seja, corresponde a estudar o conjunto $\mathbb{P} = \{x \in \mathbb{N} : x \text{ é par}\}$ e nomeadamente $\chi_{\mathbb{P}}$.

É importante notar que o conjunto $\mathcal{C}$ de todos os programas tem a cardinalidade dos naturais. Para tal basta mostrar que existe uma função injectiva de $\mathcal{C}$ para $\mathbb{N}$. Considere, por exemplo, a função injectiva γ de $\mathcal{C}$ para $\mathbb{N}$ definida recursivamente como se segue:

- $\gamma(R_i := 0) = 2^i$;
- $\gamma(R_i := R_i + 1) = 3^i$;
- $\gamma(R_i := R_j) = 5^i \times 7^j$;
- $\gamma(C_1; C_2) = 11^{\gamma(C_i)} \times 13^{\gamma(C_2)}$;
- $\gamma(\texttt{if } (R_i = R_j) \texttt{ then } C_1 \texttt{ else } C_2) = 17^i \times 19^j \times 23^{\gamma(C_1)} \times 29^{\gamma(C_2)}$;
- $\gamma(\texttt{while } (R_i \neq R_j) \texttt{ do } C) = 31^i \times 37^j \times 41^{\gamma(C)}$.

O leitor deverá ter em mente que a função γ deve ter uma função inversa simples de calcular, e por esta razão se utilizam números primos para definir γ. Salienta-se que a ideia de utilizar números primos para obter funções injectivas com inversas simples deve-se a Gödel.

Para concluir a secção sobre conceitos básicos, é necessário discutir a existência de um programa universal U e da respectiva função universal computável $f^2_U : \mathbb{N}^2 \to \mathbb{N}$ tal que

$$f_U(m,x) = \begin{cases} f^1_{\gamma^{-1}(m)}(x) & \text{se } m \in \text{img}\gamma \\ \text{indefinido} & \text{caso contrário.} \end{cases}$$

Pela definição acima, um programa universal recebe dois naturais m e x. Se m representa um programa C (isto é se $m = \gamma(C)$) então o programa U simula a execução do programa C sobre a entrada x. Caso contrário não termina. No fundo um programa universal é um simulador que recebe o código de um programa e um natural e simula a execução do programa sobre este natural. Construir um programa universal é relativamente simples, apesar de ser uma tarefa penosa. Um programador mais ou menos experiente consegue intuir que é possível ler um natural, transformá-lo num programa e depois simular as respectivas atribuições e composições de programas. Fundamentalmente, este trabalho é a tarefa do compilador conjuntamente com o esforço de executar o código compilado.

O estudo dos conceitos apresentados nesta secção deu origem a um nova área na Matemática denominada por Computabilidade.

2 Alguns conjuntos indecidíveis

A técnica utilizada por Church e Turing para falsificar o Entscheidungsproblem consistiu em encontrar um conjunto A que não é decidível, ou seja, para o qual não há nenhum programa que compute χ_A. Depois demonstraram que o Entscheidungsproblem se reduzia à decidibilidade deste conjunto A.

O conjunto proposto por Turing é denominado por K e define o Problema da Terminação de Programas:

$$K = \{(m,x) \in \mathbb{N}^2 : m \text{ representa um programa e } \gamma^{-1}(m) \text{ termina para } x\}.$$

Por outras palavras, K contém os pares de naturais para os quais a primeira componente representa um programa que termina quando recebe como entrada a segunda componente do par. Vale a pena salientar que, apesar de não o fazermos neste texto, se pode formalizar rigorosamente o Problema da Terminação de Programas em lógica de primeira-ordem.

A prova que K não é decidível é feita através de um argumento de diagonalização. Suponha-se que K é decidível e que existe um programa C_K

que computa a sua função característica. Considere-se então o seguinte programa C'_K:

C_K;
`if` $(R_1 = 1)$ `then`
$\quad R_2 = 0$; `while`$(R_1 \neq R_2)$ `do` $R_1 = R_1$
`else`
$R_1 = R_1$.

Claramente que se o programa C_K existir então também é possível desenvolver o programa C'_K. Nesse caso, seja $\ell = \gamma(C'_K)$ e note que para a entrada (m, x) o programa C'_K termina se e só se $m = \gamma(C)$ para algum C e C não termina para a entrada x.

A existência do programa C_K leva a um absurdo ao analisar se $(\ell, \ell) \in K$ ou se $(\ell, \ell) \notin K$. Por um lado, se $(\ell, \ell) \in K$ então, por definição de K, o programa $\gamma^{-1}(\ell)$ termina para a entrada ℓ. Mas como $C'_K = \gamma^{-1}(\ell)$, pela construção de C'_K, o programa $\gamma^{-1}(\ell)$ não termina para a entrada ℓ. Por outro lado, se $(\ell, \ell) \notin K$, então, por definição de K o programa $\gamma^{-1}(\ell)$ não termina para a entrada ℓ. Mas, mais uma vez, como $C'_K = \gamma^{-1}(\ell)$, então pela construção de C'_K o programa $\gamma^{-1}(\ell)$ termina para a entrada ℓ.

Desta forma pode-se concluir que o programa C_K não pode existir e portanto o Problema da Terminação de Programas é indecidível.

Na prática, isto significa que é impossível desenvolver um programa que, dado um programa C e uma entrada $x \in \mathbb{N}$, determina se a execução do programa C vai terminar para a entrada x. Note que tal programa seria uma mais valia para o desenvolvimento de aplicações informáticas. Qualquer utilizador de computadores já enfrentou situações em que o computador não responde e parece estar bloqueado (muitas vezes porque entrou em ciclo infinito e não termina). De facto, se K fosse decidível seria possível verificar se um programa termina para certas entradas. O facto de tal não ser possível torna bem mais difícil o desenvolvimento de aplicações informáticas robustas.

3 Reais não computáveis

Um facto curioso que é consequência de K ser indecidível é que se torna possível descrever um número real não computável. Um número $x \in \mathbb{R}$ é dito computável se existir um programa C tal que

$$\lim \frac{f_C^1(n)}{10^n} = x.$$

Da definição tira-se que um real x é computável se existir um programa que, dado um natural n, retorna o n primeiros algarismos significativos de x.

Como o conjunto de todos os programas tem cardinalidade $\aleph_0$ e $\mathbb{R}$ tem cardinalidade $\aleph_1$, a grande maioria dos reais não é computável. No entanto, tente o leitor indicar um único real que não seja computável. É fácil de mostrar que todos os números racionais são computáveis. Dos irracionais, é ainda fácil mostrar que todos os números algébricos[1] são computáveis. Dos números transcendentes, como π, e, e outros obtidos por funções trignométricas, hiperbólicas, entre a panóplia de funções reais estudadas durante séculos, torna-se evidente que todos os números reais que se podem descrever são computáveis. Assim sendo, como nos pode estar a escapar a esmagadora maioria de números reais, os ditos não computáveis?

Para construir um real não computável é necessário ter em mãos um conjunto indecidível. Considere ainda uma bijecção entre $\mathbb{N}^2$ e $\mathbb{N}$ como por exemplo a dada por

$$p(k,m) = 2^k(2m+1) - 1$$

cuja função inversa se representa por $p^{-1} : \mathbb{N} \to \mathbb{N}^2$. Facilmente o leitor verifica que p é computável e também o são $p_1^{-1}, p_2^{-1} : \mathbb{N} \to \mathbb{N}$ tais que $p^{-1}(n) = (p_1^{-1}(n), p_2^{-1}(n))$. Defina a seguinte sucessão

$$\begin{aligned} u_0 &= 0 \\ u_{n+1} &= \begin{cases} u_n \times 10 + 1 & \text{se } p^{-1}(n) \in K \\ u_n \times 10 & \text{caso contrário.} \end{cases} \end{aligned}$$

Note que $v_n = \frac{u_n}{10^n}$ é crescente (em sentido lato) e que $v_n < 1$, logo v_n tem limite. Como será de esperar, $x = \lim v_n$ não é um real computável, pois caso contrário, u_n seria computável o que implicaria a decidibilidade de K.

4 Conclusão

Os trabalhos de Church, Kleene e Turing, em particular a definição do Problema da Terminação de Programas por Turing, foram pioneiros para abrir uma nova área na Matemática, a Computabilidade. O leitor poderá encontrar vários livros sobre este assunto [3, 1, 9], entre os quais alguns em língua portuguesa [5, 8].

[1] Entende-se por real algébrico uma raiz real de um polinómio a uma variável com coeficientes racionais.

No presente momento, a Computabilidade é uma área de investigação estável. Trabalhos recentes têm sido realizados para entender se certas equações diferenciais relevantes têm soluções computáveis. Uma área que se desenvolveu em consequência da Computabilidade é a Complexidade Computacional [7], onde se estuda a dificuldade de problemas decidíveis em termos do tempo e espaço necessários para os decidir. A Complexidade Computacional utiliza como modelo básico de computação as máquinas de Turing. Um dos problemas do milénio, $P \neq NP$, é considerado a questão mais importante da Complexidade Computacional. Em termos de complexidade de tempo conjectura-se que certos modelos de computação implementáveis fisicamente não são equivalentes às máquinas de Turing, nomeadamente, os modelos de computação quântico, apesar de estes não computarem mais funções que as máqiunas de Turing. Outra área de investigação relevante adjacente à Computabilidade é a Complexidade de Kolmogorov, definida como o tamanho do menor programa necessário para gerar um certo número.

O legado de Turing é vasto e rico em problemas que continuarão a alimentar o esforço de investigação na área da Matemática.

Referências

[1] Douglas S. Bridges. Computability: A Mathematical Sketchbook, Spring-Verlag, 1994.

[2] Alonzo Church. *A note on the Entscheidungsproblem*, Journal of Symbolic Logic, 1 (1936), pp 40–41.

[3] Nigel Cutland. *Computability. An introduction to recursive function theory*. Cambridge University Press, 1980.

[4] N. Dershowitz. *The four sons of Penrose*. Proceedings of the Eleventh Conference on Logic Programming for Artificial Intelligence and Reasoning (LPAR), LNAI 3835, Springer, (2005), pp. 125-138.

[5] Kurt Gödel. *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I*, Monatshefte für Mathematik und Physik, 38 (1931), pp 173-98. Traduzido em Português por M. Lourenço, O teorema de Gödel e a hipótese do contínuo, 2.ª edição, revista e aumentada, 2009. F.C. Gulbenkian.

[6] John von Neumann. First Draft of a Report on the EDVAC, 1945.

[7] Christos Papadimitriou. *Computational Complexity.* Addison Wesley, 1993.

[8] Amílcar Sernadas e Cristina Sernadas. *Fundamentos de Lógica e Teoria da Computação - Segunda Edição.* College Publications, London, 2012.

[9] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing, 1997.

[10] Alan Turing. *On computable numbers, with an application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society, Series 2, 42 (1937), pp 230–265.