História da Computação: A Máquina de Turing e suas contribuições para a sociedade

Hiago Soares de Araujo

Departamento de Computação e Matemática
Universidade de São Paulo (USP) - Ribeirão Preto/SP

hiagosoares@usp.br

Abstract. *The Turing machine is a theoretical computational model whose formulation represents a milestone for Computing and affects the field to this day. In this sense, this article aims to address the history of the creation of this machine and of its concepts, as well as to explore its influence on contemporary society.*

Resumo. *A máquina de Turing é um modelo teórico computacional cuja formulação representa um marco para a Computação e afeta a área até hoje. Nesse sentido, este artigo tem como objetivo abordar a história da criação dessa máquina e de seus conceitos, além de explorar a sua influência na sociedade contemporânea.*

1. Introdução

A história da Computação é marcada por diversos personagens que contribuíram com grandes avanços para a área. Entre eles, destaca-se Alan Mathison Turing (1912 – 1954), que, além de outros importantes feitos, foi fundamental ao ajudar a estabelecer as bases da computação moderna e, devido a isso, é conhecido como o "pai da Computação".

Turing, matemático e cientista da computação britânico, por meio da formulação da Máquina de Turing, um modelo abstrato de um dispositivo computacional descrito matematicamente, contribuiu para os campos da computabilidade, teoria dos autômatos e linguagens formais e análise de algoritmos, além de formalizar matematicamente a noção de algoritmo e de investigar a extensão e as limitações do que pode ser computado.

Diante desse contexto, este artigo tem o objetivo de, evitando ao máximo notações matemáticas complexas e conceitos técnicos, e por meio da abordagem histórica, explicar os impactos da máquina desenvolvida por Turing na sociedade atual.

Figura 1. Alan Turing

2. Contexto histórico

A criação da Máquina de Turing — chamada por ele de máquina de computação — , bem como o estudo da computabilidade — campo da Computação que abrange a resolução de problemas por meio *algorítmico* e os modelos de computação —, surgem em um contexto em que várias questões fundamentais de lógica matemática estavam sendo debatidas. Mais especificamente, essas questões consistiam nos 23 problemas em aberto enunciados pelo matemático alemão David Hilbert (1862 – 1943), em um congresso internacional de matemática. Um desses problemas abordava a busca por um algoritmo para resolver equações diofantinas, e que posteriormente evoluiu para um problema mais geral, o *Entscheidungsproblem* (problema da decisão), que consistia em determinar a existência de um algoritmo que pudesse demonstrar, em tempo finito, a possibilidade ou não de se provar uma proposição matemática a partir de um conjunto de axiomas.

Foi nesse contexto que Turing, em seu artigo *"On Computable Numbers, with an Application to the Entscheidungsproblem"*, com o objetivo de dar uma resposta negativa ao problema de Hilbert, elaborou um **modelo teórico** de uma máquina que fosse capaz de formalizar o conceito de computabilidade e de resolver corretamente determinados problemas considerando a existência de um número finito e bem definido de passos para resolvê-los. Tal máquina pode ser **considerada** como a formalização matemática da noção intuitiva de algoritmo. Nesse mesmo artigo, Turing demonstrou que não existe uma máquina de Turing que resolva o *Entscheidungsproblem*.

Cabe ressaltar, no entanto, que o termo "considerada" foi utilizado porque, como será descrito mais adiante, a afirmação feita decorre da *Tese de Church-Turing*, uma tese sobre a natureza da computação que afirma, sem demonstração matemática, que **todo** problema resolvível por meio de um algoritmo pode ser computado por uma máquina de Turing, e vice-versa, de tal forma que a própria máquina é a definição formal de algoritmos.

Nesse sentido, nota-se que, ao contrário do que se poderia supor, a Máquina de Turing não surgiu dentro de um contexto puramente computacional, isto é, não

foi criada apenas com o propósito de avançar o campo da Computação. Ela foi concebida como uma ferramenta auxiliar para abordar um problema profundamente teórico da Matemática, mas que hoje estabelece as bases da computação moderna.

3. A Máquina de Turing

No artigo citado anteriormente, Turing descreveu uma máquina teórica que, seguindo passos bem definidos, pode resolver — ou mais precisamente: computar — um determinado problema. Esta seção se preocupa em dar uma definição informal e formal da referida máquina.

3.1 Definição informal

De acordo com Sipser (2005, p. 146), " O modelo da máquina de Turing usa uma fita infinita como sua memória ilimitada. Ela tem uma cabeça de fita que pode ler e escrever símbolos e mover-se sobre a fita. Inicialmente, a fita contém apenas a cadeia de entrada e está em branco em todo o restante. Se a máquina precisa armazenar informação, ela pode escrevê-la sobre a fita. Para ler a informação que ela escreveu, a máquina pode mover sua cabeça de volta para a posição onde a informação foi escrita. A máquina continua a computar até que ela decida produzir uma saída. As saídas 'aceite' e 'rejeite' são obtidas entrando em estados designados de aceitação e de rejeição. Se não entrar em um estado de aceitação ou de rejeição, ela continuará para sempre, nunca parando. ".

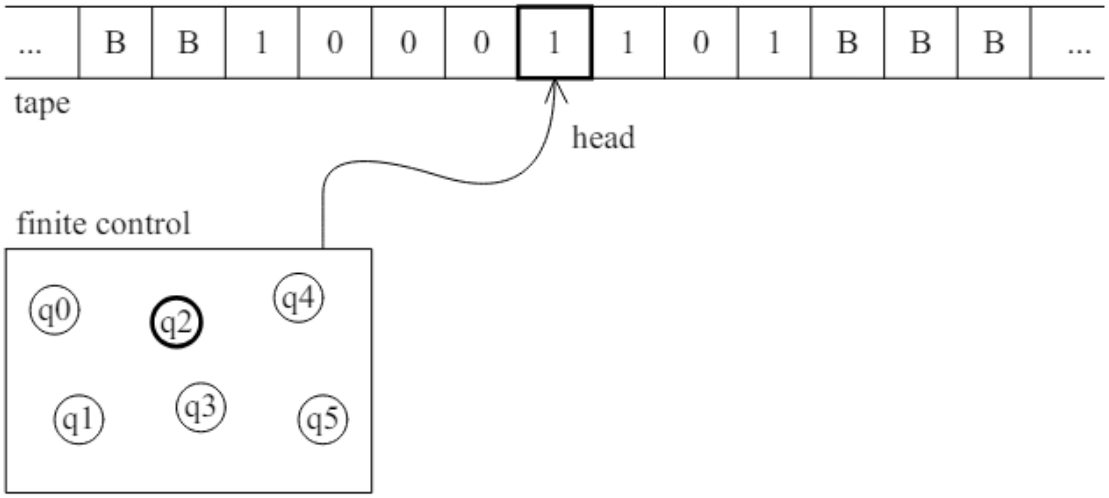

Figura 2. Máquina de Turing qualquer computando determinada entrada

3.2 Definição formal

Formalizando, podemos definir uma Máquina de Turing como sendo, de acordo com Sipser (2005, p. 148), uma 7-upla $(Q, \Sigma, \Gamma, \delta, q0, qaceita, qrejeita)$ em que Q, Σ, Γ são todos conjuntos finitos e E e D são os movimentos de leitura da fita — esquerda ou direita —, de tal forma que

- Q é o conjunto de estados,

- Σ é o alfabeto de entrada não contendo o símbolo em branco _,
- Γ é o alfabeto de fita, onde _∈ Γ e Σ ⊆ Γ ,
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$ é a função de transição,
- $q0 \in Q$ é o estado inicial,
- $qaceita \in Q$ é o estado de aceitação, e
- $qrejeita \in Q$ é o estado de rejeição, onde $qrejeita \neq qaceita$.

  A dita "cadeia de entrada" é formada por uma combinação de símbolos de Σ.

Importante destacar que o ponto central dessa definição é a função de transição δ , pois ela define os comportamentos possíveis da máquina — pode-se pensar como os passos possíveis de um procedimento —, isto é, como ela se move, sob as condições especificadas, de um estado para outro.

Exemplificando, se a máquina está em um determinado estado $q \in Q$ e a cabeça da fita está em uma célula da fita que contém o símbolo $a \in \Gamma$, o movimento $\delta(q, a) = (q', a', E)$, com $q' \in Q$, $a' \in \Gamma$, faz com que a máquina escreva o símbolo a', substituindo a , vá para o estado q' e mova a cabeça da fita para a esquerda após a substituição.

Um termo utilizado constantemente é o de *configuração* da máquina de Turing. A configuração é composta, considerando o momento instantâneo da máquina, pelo estado, o conteúdo da fita e o símbolo sendo lido, de forma que a próxima configuração é definida pela anterior.

Quando — e somente — a máquina atinge os estados $qaceita$ ou $qrejeita$, ela para e não realiza mais nenhum movimento. Considerando isso, podemos definir a função de transição δ como sendo $\delta: Q' \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$, em que $Q' = Q - \{qaceita, qrejeita\}$.

4. Contribuições

Ao longo deste texto foi destacado que a criação da Máquina de Turing representa um marco para a computação moderna. Esta seção se preocupa em mostrar as contribuições proporcionadas por essa máquina que corroboram a afirmação.

4.1 Tese de Church-Turing

De forma simplificada, a Tese de Church-Turing afirma que **todo** problema computacional, isto é, todo problema que possui solução algorítmica, pode ser descrito em termos de uma máquina de Turing que o compute. Além disso, qualquer problema que não seja possível de ser computado por uma máquina de Turing não possui nenhuma forma de ser solucionado seguindo um número finito de passos que o responda corretamente. Nesse sentido, uma importante conclusão é que, se surge uma dúvida se determinado problema pode ou não ser solucionado de forma algorítmica, basta construir uma máquina de Turing que o compute. Diante

do exposto, a ideia da Máquina de Turing como a definição formal de algoritmos é evidente.

A formulação dessa tese surgiu dos esforços independentes de Alonzo Church (1903 - 1995) e Turing ao tentarem formalizar a noção de computabilidade. Primeiro por Church, com a descrição do cálculo lambda (λ) e a suposição de que toda função efetivamente calculável — de forma simplificada, um algoritmo que sempre responda corretamente —, que transforme números inteiros em números inteiros, pudesse ser definível no cálculo-λ. Depois Turing, com sua máquina de computação e a demonstração da equivalência do seu modelo com os existentes à época, inclusive o cálculo-λ de Church.

Dessa forma, as ideias advindas da referida tese foram utilizadas por Turing para resolver o *Entscheidungsproblem*. Turing demonstrou que é impossível construir uma máquina capaz de determinar a existência de um algoritmo que pudesse demonstrar, em tempo finito, a possibilidade ou não de se provar uma proposição matemática a partir de um conjunto de axiomas. De forma que, pela tese, o algoritmo não existe.

É importante destacar, no entanto, que essa tese implica afirmações que são desprovidas de demonstração matemática, mas que são amplamente aceitas e utilizadas até os dias atuais por não existir um contraexemplo conhecido, devido a não existência de um modelo teórico computacional que seja comprovadamente mais poderoso do que o da máquina de Turing. Dessa forma, assumimos, no restante deste texto, que a tese de Church-Turing é válida.

Por fim, percebe-se que tal tese fornece uma definição clara do que é um algoritmo e é fundamental para estabelecer os limites do que pode ou não ser computado.

4.2 A máquina universal e o computador moderno

A máquina universal de Turing é uma máquina que pode simular qualquer outra máquina de Turing. Isso significa que se uma máquina for construída para computar um problema algorítmico qualquer, então esse mesmo problema pode ser computado pela máquina universal. Tal máquina possui uma interpretação semelhante a dada às máquinas de Turing em **4.1**, pelo fato de também ser uma, e pode ser compreendida, portanto, como o limite máximo de tudo o que pode ser computado. Desse modo, se existem problemas que não são possíveis de serem computados por uma máquina universal, então esses problemas são, de fato, incomputáveis.

A máquina universal de Turing nos remete a um dispositivo indispensável atualmente: a máquina de propósito geral ou, simplesmente, o computador. O computador moderno pode ser considerado, com certas limitações óbvias como a impossibilidade de se ter memória infinita (referente à fita infinita), uma implementação da máquina universal, em que cada máquina de Turing possível de

ser construída representa um algoritmo possível de ser computado pelo computador — um programa. Dessa forma, esse modelo nos permite compreender os limites, a extensibilidade e o potencial da computação moderna. Não precisamos construir, portanto, uma infinidade de computadores progressivamente complexos para descobrir se um determinado problema possui solução computacional: basta considerar as características do modelo teórico da máquina universal.

4.3 Decidibilidade e o problema da parada

Para Hilbert, nenhum problema matemático poderia ser considerado insolúvel, apenas ainda não se conhecia a solução. Ele almejava que a Matemática fosse uma ciência que evitasse o Ignorabimus — máxima em latim que expressa o sentimento de que o conhecimento científico é limitado — e, com base nisso, julgou que o *Entscheidungsproblem* fosse um problema solúvel. Turing e Church, no entanto, provaram que esse pensamento estava errado. Turing realizou esse feito demonstrando, em seu artigo, que um problema parecido com o que hoje é conhecido como o "problema da parada" não possui solução algorítmica que o responda corretamente e em tempo finito — em termos atuais, é **indecidível** —, o que o ajudou a resolver o problema de Hilbert.

Antes da explicação sobre o problema da parada, é oportuno esclarecer pontos importantes. Atualmente, considera-se que problemas "computáveis", "resolvíveis", "solucionáveis", "recursivos" são equivalentes a problemas do tipo "decidíveis". Um problema é dito ser decidível se ele pode ser solucionado em determinado número de etapas bem definidas, em tempo finito e que apresente resultados para **todo** tipo de entrada — que faça parte da resolução do problema ou não. Isso é equivalente a existir a possibilidade de se construir uma máquina de Turing do tipo **decisor** que reconheça esse problema, isto é, uma máquina que sempre pare, explicitando se a entrada é válida ou não. Alternativamente, um problema indecidível não é reconhecido por uma máquina de Turing do tipo decisor, de forma que sempre para se a entrada for válida, mas pode parar se a entrada for inválida ou permanecer em estado de loop — caso em que a resposta será desconhecida.

Ademais, cabe mencionar que a divisão de tipo das máquinas feita originalmente é diferente da estabelecida atualmente. Turing não classificou suas máquinas como sendo do tipo decisor ou não, mas sim como *circular*, que possui como requisito escrever na fita um número finito de símbolos 0 ou 1, e *circle-free*, que não possui esse requisito e nunca para. Ainda, ele define sua noção de computabilidade por meio de uma máquina *circle-free.* A noção de computabilidade estabelecida atualmente é uma adição feita à teoria de Turing por Martin David Davis (1928 - 2023) e Stephen Kleene (1909 - 1994), baseando-se em um caso particular da máquina de Turing: a de que ela sempre deva parar.

Além disso, Turing não faz menção a nenhum "problema da parada", do inglês "halting problem" — as expressões "halt" e "halting" não aparecem

nenhuma vez em seu artigo original. Tampouco menciona, em sua definição de computabilidade, a necessidade de uma máquina parar. Os problemas tratados em seu artigo, portanto, foram dois, e a demonstração da inexistência de solução algorítmica para nenhum deles serviu como base para dar uma resposta negativa à computabilidade do *Entscheidungsproblem*. O primeiro problema consistia em determinar se uma determinada máquina era circular ou circle-free. O segundo, que foi o utilizado de fato na resolução do problema de Hilbert e é conhecido como PRINT?, consistia em determinar se alguma vez uma máquina iria escrever na fita determinado símbolo. Ao utilizar esse problema para resolver o *Entscheidungsproblem*, Turing realiza um feito fundamental ao demonstrar que existem problemas que não podem ser solucionados em um número finito e bem definido de etapas e contribui para estabelecer os limites da Computação.

Nesse sentido, o problema hoje conhecido como “problema da parada” foi, na verdade, formulado, popularizado e demonstrado ser indecidível por Davis em seu livro *Computability and Unsolvability* (1958). Ele utilizou como referência o PRINT? de Turing para a sua abordagem.

Ressalvas históricas feitas, pode-se agora explicar o referido problema. O problema da parada consiste em decidir se uma máquina de Turing qualquer — ou, similarmente, um programa de computador —, sob determinada entrada, irá parar ou não. Davis demonstrou que esse problema é indecidível e, portanto, não possui solução algorítmica. A negativa ao problema da parada é amplamente utilizada para demonstrar que determinados problemas não podem ser computados. A técnica utilizada, nesses casos, é reduzir o problema à resolução do problema da parada que, como mencionado, não possui solução. Tal fato implica, mais uma vez, que existe um limite em relação ao que pode ou não ser solucionado de forma computacional.

4.4 Teoria dos Autômatos e Linguagens Formais

Considerando as características da máquina de Turing, pode-se desenvolver máquinas que sigam modelos computacionais mais limitados. Essas máquinas, chamadas *autômatos*, são utilizadas especificamente para resolver determinados problemas que são representados em forma de *linguagens formais*. Na hierarquia de modelos de computação, os autômatos são classificados em diferentes níveis de poder computacional, e cada tipo de autômato é capaz de reconhecer uma classe específica de linguagens formais, que representam os problemas que ele é capaz de computar. Essas linguagens são geradas por gramáticas que são classificadas de acordo com o que hoje é conhecido como a Hierarquia de Chomsky. Autômatos de classes específicas são utilizados, atualmente, na descrição de linguagens de programação e em ferramentas como interpretadores e compiladores.

4.5 Linguagens de programação Turing completas

Desconsiderando particularidades de hardware e considerando apenas aspectos teóricos, uma linguagem de programação é considerada Turing completa se ela

pode simular o comportamento de qualquer máquina de Turing. Isso significa que ela possui o poder computacional de uma máquina universal. Em termos práticos, uma linguagem de programação Turing completa pode ser utilizada para codificar qualquer problema possível de ser computado. A maioria das linguagens atuais é Turing completa.

5. Conclusão

Neste trabalho, foi possível explorar uma parte fundamental da história da Computação por meio de uma das mais importantes contribuições de Alan Turing à sociedade: a máquina de Turing. Analisando o contexto histórico de sua criação, foi destacado como as inquietações matemáticas do início do século XX levaram Turing a desenvolver aquilo que posteriormente seria considerado um dos pilares da computação moderna.

Ainda, foram abordadas as diversas contribuições dessa máquina. A Tese de Church-Turing fornece a possibilidade de formalizar a noção intuitiva de algoritmo e auxilia a compreender os limites do que pode ser computado. A ideia de uma máquina universal, capaz de simular qualquer outra máquina de Turing, reforça os limites da computação e pavimenta o caminho do desenvolvimento dos computadores modernos. Além disso, a demonstração de que o *Entscheidungsproblem* não possui solução, a extensão à teoria de Turing com as máquinas do tipo decisor, o conceito de decidibilidade bem como o problema da parada nos mostram como provar que existem problemas que não podem ser solucionados computacionalmente. Destaca-se, também, o impacto nas linguagens de programação e o surgimento de outros campos da Computação relacionados à máquina de Turing.

Conclui-se, portanto, que a referida máquina, embora concebida em um contexto teórico e matemático, transcendeu suas origens para se tornar um pilar fundamental da Computação moderna. Suas contribuições influenciaram não apenas a teoria, mas também a prática dessa área. Entender sua história, seus conceitos e impactos é essencial para compreender o estado atual do desenvolvimento da Computação bem como as possibilidades de avanço para o futuro.

Referências

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem.
https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf

De Mol, Liesbeth, "Turing Machines", The Stanford Encyclopedia of Philosophy (Winter 2021 Edition), Edward N. Zalta (ed.).
https://plato.stanford.edu/archives/win2021/entries/turing-machine

Copeland, B. Jack, "The Church-Turing Thesis", The Stanford Encyclopedia of Philosophy (Spring 2024 Edition), Edward N. Zalta & Uri Nodelman (eds.). https://plato.stanford.edu/archives/spr2024/entries/church-turing

Immerman, Neil, "Computability and Complexity", The Stanford Encyclopedia of Philosophy (Winter 2021 Edition), Edward N. Zalta (ed.). https://plato.stanford.edu/archives/win2021/entries/computability

Hopcroft, J. (2012). On the Impact of Turing Machines. In: Agrawal, M., Cooper, S.B., Li, A. (eds) Theory and Applications of Models of Computation. TAMC 2012. Lecture Notes in Computer Science, vol 7287. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29952-0_1

Salvador Lucas, The origins of the halting problem, Journal of Logical and Algebraic Methods in Programming, Volume 121, 2021, 100687, ISSN 2352-2208, https://doi.org/10.1016/j.jlamp.2021.100687

LOFF, Bruno. A tese de Church–Turing. Sociedade Portuguesa de Matemática, 2012. https://revistas.rcaap.pt/boletimspm/article/view/3870

Vitányi, Paul. (2009). Turing Machines and Understanding Computational Complexity https://www.researchgate.net/publication/220579932_Turing_machine

Old Dominion University. Turing Completeness CS390, Summer 2024. https://www.cs.odu.edu/~zeil/cs390/latest/Public/turing-complete/index.html