A tese de Church–Turing

Bruno Loff
Centrum voor Wiskunde en Informatica
Science Park 123
1098XG Amsterdão, Holanda
e-mail: bruno.loff@gmail.com

Resumo: De acordo com a tese de Church–Turing, se um cálculo puder ser feito de forma automatizada — por um dado método, num número finito de passos — então também pode ser feito por uma máquina de Turing. Neste artigo faz-se uma breve introdução à tese de Church–Turing e ao contexto histórico da sua formulação. Inclui-se uma tradução comentada de parte do artigo de Alan Turing de 1936–37, "On computable numbers, with an application to the Entscheidungsproblem" [24], onde é possível entender a origem da máquina de Turing.

Abstract The Church–Turing thesis roughly states that, if a certain computation can be carried out in an automated fashion, by some calculation procedure and in a finite number of steps, then it can also be carried out by a Turing machine. In this article I will make a brief introduction to the Turing thesis, and to the surrounding historical context. Also included is a commented partial translation, to Portuguese, of the relevant section of Turing's original 1936–37 article "On computable numbers, with an application to the Entscheidungsproblem" [24], where it is possible to understand how the Turing machine came to be.

palavras-chave: Computabilidade, Tese de Church–Turing, Máquina de Turing

keywords: Computability, Church–Turing Thesis, Turing Machine

Preâmbulo: análise do algoritmo da soma

1. Este artigo serve dois propósitos. Primeiro, pretende explicar o melhor possível, a um leitor com pouco conhecimento sobre o tema, o que diz a tese de Church–Turing. Assim, começamos nesta secção por analisar o algoritmo para a soma de dois números, que é o mais simples que todos conhecemos; depois, na quarta e última secção do artigo, incluímos a tradução comentada de uma parte do artigo original de Turing, onde é feita uma análise semelhante para o caso geral de um algoritmo arbitrário; nesta secção procuramos mostrar como a definição de máquina de Turing surge

muito naturalmente desta análise. O segundo propósito deste artigo é dar uma visão global dos acontecimentos históricos que levaram à formulação da tese de Church–Turing. Deste modo, na segunda secção do artigo descrevemos o contexto lógico-matemático do princípio do século XX, que motivou o estudo da computabilidade. Na terceira secção narramos os acontecimentos que resultaram numa definição satisfatória da mesma, e na formulação da tese de Church–Turing.

2. Esta tese, ainda que pouco conhecida, pertence a um conjunto de descobertas, feitas nos anos 30 do século passado, que influenciaram drasticamente a nossa história e o nosso modo de vida. A tese diz respeito ao conceito de algoritmo, que no contexto deste artigo definimos da seguinte forma:

> **Algoritmo.** Procedimento para resolver um problema matemático num número finito de passos.

Vamos então olhar com detalhe para o algoritmo escolar para a soma de dois números.

3. Suponhamos que nos são dados dois números a e b, em notação decimal, e pretendemos calcular o número $a + b$. Os números são escritos de tal forma a que o dígito à direita de a está alinhado com o dígito à direita de b, e cada um dos seguintes dígitos de a está alinhado com o dígito de b que lhe corresponde. Procedemos então da direita para a esquerda somando uma casa decimal de cada vez, com atenção ao transporte ("e vai um"). Podemos adicionar cada par de dígitos de duas formas: utilizando os dedos das mãos como uma criança, ou relembrando uma tabela com as somas de pares de dígitos de 0 a 9. O dígito resultante desta soma é escrito abaixo do respectivo dígito de b. Depois de completar o cálculo para cada dígito dos dois números, o número $a+b$ estará escrito abaixo de b em notação decimal.

Assim descrito, este algoritmo tão simples parece um pouco mais complicado. Mas ainda não dissecámos o algoritmo completamente; para tal temos de descrever com exactidão cada operação individual.

4. Vamos então estabelecer as convenções seguintes. Suponhamos que nos é dada uma folha de papel quadriculado suficientemente grande. Os números a e b estão escritos em notação decimal, em duas linhas adjacentes, alinhados à direita. Suponhamos adicionalmente que só podemos focar a nossa atenção em três quadrículas de papel de cada vez, alinhadas verticalmente. Por exemplo veja-se a Figura 1, para o caso em que a é cento e cinquenta e seis e b é quarenta e sete. Tal como na figura, vamos supor que no princípio do procedimento a nossa atenção está focada nas quadrículas à direita.

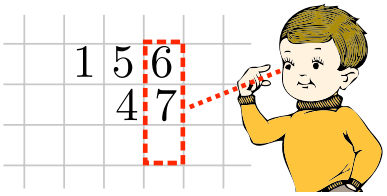

Figura 1: A configuração inicial do algoritmo da soma.

5. Durante a execução do algoritmo, o nosso "estado mental" determina as operações que pretendemos executar. A soma de dois números é feita pela soma sucessiva de cada par de dígitos, e para somar um par de dígitos correctamente, é necessário saber se houve transporte de uma unidade vinda da soma do par de dígitos anterior. Portanto é possível estar em dois *estados mentais* distintos: um estado mental no qual não há transporte, e um estado mental no qual há transporte. Vamos chamar a estes estados mentais *soma de um dígito sem transporte* (S) e *soma de um dígito com transporte* (T). O nosso propósito é diferente em cada estado mental. Se estamos no estado mental S, o nosso objectivo é somar o dígito na quadrícula de cima (do nosso foco de atenção) com o dígito na quadrícula do meio e escrever o resultado na quadrícula de baixo. O nosso objectivo no estado T é o mesmo, mas acrescentando 1 ao resultado.

O algoritmo começa, como já foi mencionado, com a nossa atenção focada nos três quadrículas da direita. Começamos por estar no estado mental S. Sempre que estivermos no estado mental S consultamos a Tabela 1(a), já memorizada, que contém todas as somas de dois dígitos, para encontrar a entrada correspondente aos dois dígitos no nosso foco de atenção. Depois escrevemos, na quadrícula de baixo, o dígito da direita do número extraído da tabela. Em seguida movemos o nosso foco de atenção para as três casas imediatamente à esquerda. Por fim, decidimos o nosso novo estado mental da seguinte forma: se o número extraído da tabela for menor que 10, então continuamos a soma dos restantes dígitos no estado mental S; e se for pelo menos 10, então continuamos no estado T. Por exemplo, se estivermos no estado mental S e virmos a seguinte configuração:

8
5

,

Então de acordo com a coluna 8 e a linha 5 da Tabela 1(a), o resultado é 13, e portanto mudamos o conteúdo das quadrículas para:

$$\begin{array}{|c|}\hline 8 \\ \hline 5 \\ \hline 3 \\ \hline \end{array},$$

movemos o nosso foco de atenção para as quadrículas imediatamente à esquerda, e porque o resultado é pelo menos 10, mudamos para o estado mental T.

No estado mental T as nossas acções são as mesmas, mas utilizando a Tabela 1(b). Se num dado momento focamos a nossa atenção em três quadrículas em branco, então chegámos ao fim dos nossos números; se nesse instante estivermos no estado T, então colocamos um 1 na quadrícula de baixo, e completámos a soma; se estivermos no estado S, então o cálculo termina sem mais operações.

Tabela 1: Tabelas para a soma; "□" representa uma quadrícula em branco.

(a) soma de um dígito sem transporte

	□	**0**	**1**	**2**	**...**	**9**
□	□	0	1	2	...	9
0	0	0	1	2	...	9
1	1	1	2	3	...	10
2	2	2	3	4	...	11
			⋱			
9	9	9	10	11	...	18

(b) soma de um dígito com transporte

	□	**0**	**1**	**2**	**...**	**9**
□	1	1	2	3	...	10
0	1	1	2	3	...	10
1	2	2	3	4	...	11
2	3	3	4	5	...	12
			⋱			
9	10	10	11	12	...	19

6. Agora a descrição do algoritmo está completa. O acto de somar dois números foi reduzido a passos distintos e indivisíveis: (1) leitura e escrita de símbolos; (2) mudança do foco de atenção; (3) mudança do estado mental.

A tese de Church–Turing diz, *grosso modo*, que qualquer algoritmo, isto é, qualquer *procedimento para resolver um problema matemático num número finito de passos,* pode ser reduzido a estes mesmos passos elementares (1), (2), e (3). Depois da análise que fizemos do algoritmo da soma, a tese de Church–Turing parece extremamente verosímil: tornou-se razoável imaginar que qualquer algoritmo, dissecado da mesma forma, poderá reduzir-se às mesmas operações. Esta observação simples, fundamental — e *a priori* nada trivial — foi feita pela primeira vez por Alan Turing, nos anos 30 do século passado.[1]

[1]No entanto, como veremos, Alonzo Church já tinha feito uma observação semelhante a respeito do cálculo-λ, um sistema matemático de sua invenção. No entanto, os passos elementares do cálculo-λ são muito mais complexos e muito menos intuitivos que as operações (1), (2) e (3) postuladas por Turing.

A noção de algoritmo no princípio do Séc. XX

7. Para compreender a tese de Church–Turing é necessário voltar um pouco atrás no tempo, para uma época antes da invenção do computador digital. No princípio do século XX quase todas as contas eram feitas ou à mão, ou mentalmente ou com a ajuda de um mecanismo simples como um ábaco ou uma régua de cálculo. Os dispositivos de cálculo mais sofisticados existentes na altura eram as máquinas de tabulação, que permitiam processar uma grande quantidade de dados em pouco tempo[2], e o analisador diferencial, que era capaz de calcular soluções numéricas de equações diferenciais simples[3].

Hoje em dia a palavra inglesa "computer" é sobretudo utilizada para designar um dispositivo electrónico, mas em 1936 um "computer" era uma pessoa contratada para fazer cálculos longos e repetitivos, típicamente num observatório astronómico. Em Português as palavras correspondentes diferem; uma máquina que faz cálculos é um "computador", mas uma pessoa que faz cálculos é um "calculador". O trabalho de um calculador, em princípio, consistia na execução de um conjunto de métodos de cálculo bem definidos.[4]

8. Já para os matemáticos da época era claro que um método de cálculo, ainda que correcto em principio, não era necessariamente aplicável na prática.

Por exemplo, para produzir as soluções inteiras da equação de Pell[5]:

$$x^2 = ay^2 + 1$$

é possível utilizar um método conhecido no ocidente desde o século XVII, e que no século XVIII Lagrange demonstrou ser correcto; todavia, em certos casos particulares os cálculos tornavam-se demasiado extensos, e o método era impossível de aplicar na prática.[6] Portanto havia já uma noção, ainda um tanto vaga, do que hoje chamamos de algoritmo.

[2]O exemplo mais famoso é o cálculo do censo dos EUA, em 1890, que demorou apenas um ano: compare-se com os sete anos necessários para concluir o censo de 1880.

[3]Utilizado, por exemplo, no cálculo de tabelas de artilharia.

[4]No início do século XX os calculadores eram muitas vezes mulheres, em parte porque as posições científicas estavam reservadas aos homens, e talvez também em parte porque na altura uma calculadora recebia metade do salário de um calculador homem. Várias destas mulheres encontraram na carreira de calculador a sua única possibilidade de fazer ciência, e algumas foram mais tarde reconhecidas como cientistas brilhantes. Veja-se o exemplo da calculadora–astrónoma Henrietta Swan Leavitt.

[5]Na equação que se segue, a é uma dada constante inteira, e procura-se todos os pares de números inteiros x e y que a satisfaçam.

[6]O artigo [14] contém uma introdução à equação de Pell, aos algoritmos para a resolver e ao *problema dos bovinos,* que foi inventado por Arquimedes e que só pôde ser finalmente resolvido mais de vinte séculos depois por um computador digital.

9. A equação de Pell é um exemplo de uma equação diofantina, isto é, uma equação da forma $p(x_1, \ldots, x_k) = 0$, para um dado polinómio p com coeficientes inteiros, em que as variáveis $x_1, \ldots, x_k$ só podem assumir valores inteiros. Na altura só eram conhecidos métodos para resolver certos casos particulares de equações diofantinas, como a equação de Pell. No virar do século XIX, o matemático alemão David Hilbert publicou, no seguimento do congresso internacional de matemáticos, uma lista de 23 problemas em aberto [10]. O décimo problema consistia na busca de um algoritmo geral para resolver equações diofantinas. A formulação original de Hilbert é a seguinte [10]:

> **10. Resolubilidade de uma equação diofantina.** Dada uma equação diofantina com um qualquer número de incógnitas e com coeficientes inteiros: inventar um processo que permita determinar, num número finito de operações, se a equação tem solução nos números inteiros[7].

Ora, *um processo que permita determinar a solubilidade de uma equação num número finito de operações* é justamente um algoritmo, de acordo com a definição que foi dada no princípio deste artigo. Nos anos que se seguiram à sua formulação, o décimo problema de Hilbert evoluiu para um problema mais geral, que ficou conhecido como o *Entscheidungsproblem* (ou *problema da decisão*) de Hilbert [6, p. 146]. Informalmente, o problema da decisão consiste na procura de um procedimento para determinar, num número finito de passos, se uma dada formula da lógica de primeira ordem é válida [11]:

> **Problema da decisão.** O Entscheidungsproblem [ou problema da decisão para a lógica de primeira ordem] pede um procedimento que permita, dada uma expressão lógica, determinar a sua validade ou satisfatibilidade num número finito de passos. Este problema deve ser considerado o problema mais importante da lógica matemática... A solução deste problema é de uma importância fundamental para a teoria de todos os domínios cujas proposições podem ser estudadas com base num número finito de axiomas.

10. De certo modo, o *Entscheindungsproblem* pedia um algoritmo mestre, que fosse capaz de demonstrar qualquer proposição matemática que pudesse

[7]Este problema na sua formulação original só viria a ser resolvido 70 anos depois, quando Yuri Matiyasevich demonstra uma conjectura de Martin Davis, Julia Robinson e Hilary Putnam, a qual implica a inexistência do dito processo [5].

ser formalizada com precisão. E na altura era muito razoável pensar que um tal algoritmo deveria mesmo existir. Nos séculos anteriores as disciplinas da matemática e da física tinham sofrido um enorme desenvolvimento e encontrado aplicações variadas. O movimento era de expansão, e havia na altura poucos resultados que mostrassem limitações gerais de métodos matemáticos. (Um exemplo clássico de um resultado deste género foi a demonstração por Ferdinand von Lindemann da transcendência de π e, como consequência, da impossibilidade da quadratura do círculo.) Na pequena comunidade que trabalhava nos fundamentos da matemática, era prevalente a crença de que o *Entschendungsproblem* seria resolvido afirmativamente.[8]

Mas em 1936, Alonzo Church e Alan Turing mostraram que um tal algoritmo mestre não existe, e para esta demonstração foi necessário encontrar uma definição precisa de *algoritmo...*

A tese de Church–Turing e suas generalizações

11. Entre 1932 e 1936 foram publicadas quatro definições formais de "computabilidade," que mais tarde se demonstrou serem equivalentes: o cálculo-λ de Alonzo Church [e.g. 3], as funções recursivas de Kurt Gödel [13], a máquina de Turing [24] e a máquina de Post [18].

O cálculo-λ foi inventado por Alonzo Church em Princeton no ano de 1932, como parte de um trabalho de fundamentos da matemática. Church encarregou os seus dois alunos de doutoramento, Stephen Kleene e John Rosser, de desenvolver esse trabalho [1]. A tese de doutoramento de Kleene e o trabalho de Rosser que se seguiu mostrou que o cálculo-λ tinha um enorme poder expressivo. John Rosser relata a conversa em que Church enunciou pela primeira vez um esboço da (actualmente chamada) tese de Church–Turing [19]:

> «Certo dia, no fim de 1933, estava a descrever [a Alonzo Church] a minha mais recente [demonstração de que uma dada função é definível no cálculo-λ]. Ele comentou, de forma pouco convicta, que talvez toda a função efectivamente calculável, que transforme números inteiros em números inteiros, possa ser definível no cálculo-λ. De facto, deu-me a impressão que essa ideia lhe tinha surgido nesse preciso momento, ao ver a minha demonstração».

[8] A expressão máxima desta ideia surge nas duas frases finais de uma aula que Hilbert deu em 1930: *Wir müssen wissen. Wir werden wissen.* — (referindo-se às verdades matemáticas) Nós temos que saber. Nós vamos saber. [6, p. 103].

12. Em 1934, chegado a Princeton, Gödel leccionou um curso onde, entre outros temas, se debruçou sobre os seus teoremas de incompletude e sobre a teoria das funções recursivas[9]. Gödel sabia que uma definição satisfatória de computabilidade lhe permitiria obter uma noção precisa de *sistema formal* e, consequentemente, um melhor entendimento de quais os sistemas formais que padeciam da incompletude por ele descoberta[10]. Para além disso, Church estava convencido que uma definição precisa de computabilidade ajudaria a entender o *Entscheidungsproblem* de Hilbert [9]. Church partilhou as suas conjecturas a respeito do cálculo-λ com Gödel. Todavia, Gödel considerava como sendo "completamente insatisfatória" esta versão embrionária da tese de Church[11].

No ano de 1935, Kleene e Rosser demonstram a equivalência entre as funções recursivas de Gödel e o cálculo-λ, e isto convence finalmente Church da validade da sua tese [20].[12] No mesmo ano Church inicia a escrita do seu artigo *An unsolvable problem in elementary number theory* [3] que resolve negativamente o Entscheindungsproblem, e que seria publicado no princípio de 1936.

13. Até este ponto, não parece haver motivo para associar o nome de Turing à tese de Church, mas de facto Turing teve um papel essencial na sua fundamentação. Trabalhando independentemente do grupo de Princeton, Alan Turing, ainda estudante de licenciatura, publica em 1936 o seu artigo *On computable numbers, with an application to the Entscheidungsproblem.* Nesse artigo Turing apresenta um dispositivo conceptual, um modelo matemático de computação. Turing mostra a expressividade do seu modelo implementando vários algoritmos sofisticados, e demonstrando a equivalên-

[9] Teoria inspirada na sua breve correspondência com Jacques Herbrand [21].

[10] Ver [23], para uma boa introdução a sistemas formais e aos teoremas da incompletude de Gödel.

[11] Como descrito no artigo [20], Gödel achava que a única forma de demonstrar a tese de Church seria partir de uma axiomatização intuitiva dos princípios que uma qualquer definição de computabilidade deva obedecer, e depois derivar um teorema mostrando como o cálculo-λ ou as funções recursivas resultam unívocamente desses axiomas. Todavia, o artigo de Turing, que apresenta uma argumentação informal — justamente na secção que traduzimos no fim deste artigo — convenceu-os ambos de que haviam chegado à definição correcta. Esse trabalho de axiomatização foi feito *a posteriori*, em diversos contextos e por diferentes autores [e.g. 8, 22, 7].

[12] A primeira apresentação pública da tese de Church surge numa palestra feita à American Mathematical Society, em 19 de Abril de 1935 [2]. No resumo desta palestra pode ler-se: *«Defendemos que a noção de uma função efectivamente calculável sobre os números inteiros deve ser identificada com a noção de função recursiva, sendo que outras definições plausíveis de calculabilidade efectiva resultam em noções que são ou equivalentes ou mais fracas que a noção de função recursiva».*

cia com outras possíveis noções, tal como Church, Kleene e Rosser tinham feito com o cálculo-λ e as funções recursivas. A máquina de Turing surge da sua análise sistemática da actividade de cálculo, que é reduzida às três operações elementares (1), (2) e (3) da secção 6. É esta análise que traduzimos para Português na secção seguinte, porque é ela que justifica a equivalência entre a máquina de Turing e o conceito intuitivo de algoritmo, e é devido a ela que se tornou costume associar o nome de Turing à tese inventada por Church.

Para além destas contribuições, Turing inventa um conceito fundamental inteiramente novo — a sua máquina universal. Turing mostrou que era possível construir uma máquina de Turing programável, capaz de simular qualquer outra máquina de Turing quando dada um programa para a descrever. Ou seja, foi o inventor do computador programável moderno, com a sua distinção entre hardware e software.[13]

14. Uma máquina de Turing é um dispositivo meramente conceptual capaz de executar as três operações elementares da §6. A máquina possui uma *fita* quadriculada, e nessa fita há uma *cabeça de leitura* que permite ler e escrever certos símbolos, e que se pode deslocar para a esquerda ou para a direita. Num dado instante da computação, a cabeça de leitura está sobre uma dada quadrícula da fita, e a máquina encontra-se num certo *estado interno* (e uma dada máquina só pode estar num estado de entre um número finito de estados). A operação feita pela cabeça de leitura, e o estado interno no instante seguinte, são determinadas pelo estado interno da máquina e pelos símbolos lidos no instante presente. A simplicidade do modelo permite chegar a uma definição formal e matemáticamente precisa, que esboçaremos na quarta parte deste artigo.

15. E agora estamos prontos para formular a tese adequadamente:

> **Tese de Church–Turing.** Toda a função que pode ser calculada por um procedimento finito pode também ser computada por uma máquina de Turing.

Uma implicação da tese de Church–Turing é que a noção de algoritmo — até então heurística e imprecisa — encontra, através da máquina de Turing, uma caracterização matemática completa. Isto permite-nos entender melhor os limites do calculável. Por exemplo, no seu artigo Turing demonstra

[13] Mais tarde na sua carreira Turing desempenha um papel essencial na decifração dos códigos utilizados nas comunicações militares da Alemanha nazi [5]; depois da Segunda Guerra Mundial publica o primeiro artigo matemático sobre inteligência artificial [26, 27, 4]. Todas estas contribuições científicas influenciaram drásticamente o curso da humanidade. A sua morte precoce foi uma grande perda.

que o *Entscheidungsproblem* de Hilbert não pode ser solucionado por uma máquina de Turing. Mas ao aceitar a tese de Church–Turing, somos forçados a concluir que esse mesmo problema não pode ser resolvido por nenhum método de cálculo; portanto, o algoritmo que Hilbert procurava não existe.

16. A nossa pequena descrição histórica evidencia que o estudo da computabilidade surgiu a partir de considerações matemáticas fundamentais e abstractas, e não de uma análise matemática inspirada em avanços tecnológicos ou em experiências cientificas. Portanto, a invenção do computador tem uma história singular: o objecto matemático já existia e estava perfeitamente caracterizado vinte anos antes da sua implementação tecnológica.

17. Nas décadas que se seguiram, os computadores electrónicos mudaram o mundo. A palavra "computação" já só raramente significa uma actividade feita por seres humanos e passou a designar a actividade feita por dispositivos electrónicos. De facto, o computador substituiu e superou o ser humano em muitas actividades ditas *inteligentes,* como jogar xadrez. A potência e a versatilidade dos computadores é tal que hoje em dia são utilizados para simular certos sistemas e experiências em engenharia, meteorologia, biologia, química e física. Um exemplo em biologia é o uso dos computadores na previsão do enovelamento de proteínas.

A tese de Church–Turing pode ser generalizada tendo em conta estas capacidades do computador. Neste artigo não nos alongaremos muito neste tema, que é nebuloso e controverso; vamos apenas enumerar alguns exemplos.

Um destes surgiu no verão de 1956, durante o evento fundador da inteligência artificial como área cientifica: a conferência de Dartmouth [16].

> **Tese de Dartmouth.** Todas as características e capacidades da aprendizagem e da inteligência podem ser efectivamente simuladas por uma máquina [de Turing].

A tese de Church–Turing da §15 é um caso particular da tese de Dartmouth, visto que a capacidade de fazer cálculos é uma das capacidades da inteligência. A tese de Dartmouth implica em particular que a inteligência humana é o resultado de um processo meramente mecânico, ou pelo menos mecanizável. Uma versão ainda mais forte é a seguinte tese [17, 15]:

> **Tese da simulação.** Todo o sistema físico pode ser simulado por uma máquina de Turing.

Ou seja, a tese da simulação propõe que o universo se rege por leis que podem ser descritas algoritmicamente. Que pensar disto? Por um lado, parece ousado atribuir ao conceito de computabilidade tal significado cósmico;

por outro lado, e vendo bem, a tese da simulação não é mais do que uma versão reforçada da crença segundo a qual o universo pode ser descrito por leis matemáticas[14].

A argumentação de Turing

18. No seu artigo de 1936, Turing faz a seguinte formulação da tese:

> «[The numbers computable by a Turing machine] include all numbers which could naturally be regarded as computable».

Em defesa desta tese, Turing oferece três linhas de argumentação: (1) mostra que certos algoritmos bastante sofisticados podem ser implementados numa máquina de Turing, o que revela a versatilidade e a riqueza do modelo; (2) demonstra a equivalência entre a máquina de Turing e outras duas noções (um cálculo funcional baseado em lógica de primeira ordem, e o cálculo-λ[15]); e (3) apresenta uma análise fenomenológica do acto de cálculo.

19. Na minha opinião, esta terceira linha de argumentação é aquela que melhor justifica a máquina de Turing e a tese de Church–Turing. Com efeito, esta argumentação explica porque é que a máquina de Turing é um modelo natural de computação, e como a definição formal de máquina de Turing surge, muito naturalmente, a partir de uma análise rigorosa do cálculo tal como um ser humano o faz. Hoje em dia considero que esta parte do artigo de Turing é leitura indispensável num curso de introdução à teoria da computação.

20. Sendo assim, permitimo-nos traduzir aqui parcialmente a secção (9a) do artigo [24]. Em paralelo, fazemos notar que a análise de Turing foi aplicada na nossa apresentação do algoritmo da soma, e como surge de forma transparente a definição formal de máquina de Turing.

[p. 249] *«Normalmente, um cálculo é feito escrevendo certos símbolos em papel. Podemos supor que este papel está dividido em quadrículas como num caderno escolar de aritmética. Em aritmética elementar faz-se, por vezes, uso do carácter bidimensional do papel quadriculado. Mas é sempre possível evitar tal uso,*

[14] A tese da simulação pode ser de novo generalizada para exigir que a simulação seja *eficiente*. Neste ponto parece ser necessário utilizar um modelo de computação quântica [ver 12].

[15] À data da sua primeira submissão, Turing desconhecia o trabalho de Church. A demonstração de equivalência com o cálculo-λ é apenas esboçada em apêndice, e mais tarde demonstrada rigorosamente no artigo [25].

> *e penso que é podemos concordar que tal uso não é essencial para o cálculo. Portanto, assumirei que o cálculo é feito num pedaço de papel unidimensional, ou seja, numa fita quadriculada. Vou também assumir que o número de símbolos que podem ser impressos é finito. Se permitíssemos um número infinito de símbolos, haveria necessariamente certos símbolos que difeririam entre si numa extensão arbitrariamente pequena*[†]*. O efeito desta restrição do número de símbolos não é muito grave. É sempre possível utilizar uma sequência de símbolos no lugar de um símbolo. Desta forma um numeral arábico, como por exemplo 17* [p. 250]
> *ou 999999999999999, é normalmente tratado como um símbolo único. Da mesma forma em qualquer língua europeia as palavras são tratadas como símbolos singulares (enquanto a língua Chinesa procura ter uma infinidade enumerável de símbolos). Do nosso ponto de vista, a diferença entre símbolos singulares e compostos é que os símbolos compostos, se forem demasiado longos, não podem ser observados num só relance. Isto está de acordo com a nossa experiência. Num só relance não nos é possível saber se 9999999999999999 e 999999999999999 são iguais.»*

Como vimos, Turing começa por analisar o modo como se fazem cálculos: escrevendo símbolos em papel. Pressupõe, por consenso, que o caracter bidimensional do papel é desnecessário. Como tal, uma máquina de Turing é um dispositivo que faz uso de uma fita quadriculada unidimensional. Em cada quadrícula da fita pode estar escrito um símbolo. A máquina possui um mecanismo, que chamamos *cabeça de leitura*, que pode ler e escrever símbolos nas quadrículas desta fita. Os símbolos a ser utilizados formam um conjunto, que designamos por Σ.

Depois, Turing estabelece que os símbolos utilizados são em número finito. Formalmente, isto significa que Σ é um conjunto finito. Como vimos, no caso do algoritmo da soma só são necessários dez símbolos: $\Sigma = \{0, 1, \ldots, 9\}$. Neste algoritmo, fizemos de facto uso do carácter bidimensional do papel. Todavia, é possível transformar o algoritmo de modo a funcionar numa

[†] «Se considerarmos que um símbolo é literalmente impresso numa quadrícula, podemos supor que essa quadrícula é $0 \leqslant x \leqslant 1$, $0 \leqslant y \leqslant 1$. O símbolo é definido como um conjunto de pontos na quadrícula, nomeadamente o conjunto ocupado pela tinta impressa. Se nos limitarmos a conjuntos mensuráveis de pontos, podemos então definir a "distância" entre dois símbolos como sendo o custo de transformar um símbolo noutro, em que o custo de deslocar uma unidade de área de tinta uma unidade de distância é a unidade, e somos providos de uma quantidade infinita de tinta no ponto $x = 2, y = 0$. Com esta topologia os símbolos formam um espaço pré-compacto.»

fita unidimensional. Deixamos para o leitor o exercício de descrever essa transformação.

21. Na parte que se segue Turing analisa o que se passa dentro da cabeça do calculador.

> *«O comportamento de um calculador em qualquer instante é determinado pelos símbolos que ele está a observar, e pelo seu "estado mental" nesse momento. Podemos supor que existe um limite B do número de quadrículas (e portanto símbolos) que o calculador pode observar num dado momento. Se ele desejar observar mais quadrículas, terá de fazer uso de observações sucessivas. Vamos supor também que o número de "estados mentais" que precisamos de levar em conta é finito. As razões para isto são do mesmo carácter que aquelas que restringem o número de diferentes símbolos. Se admitíssemos uma infinidade de estados mentais, alguns destes iriam estar "arbitrariamente próximos," e seriam confundidos. De novo, a restrição não é uma que afecte seriamente o cálculo, dado que o uso de estados mentais mais complicados pode ser evitado escrevendo mais símbolos na fita.»*

A simplificação feita é drástica e certeira. As acções julgadas relevantes para o cálculo dependem apenas dos símbolos observados e do "estado mental" do calculador. Os símbolos observados são em número finito, sendo claro para qualquer pessoa que é impossível observar uma infinidade de símbolos. Por outro lado, não importa qual o conteúdo ou a função dos "estados mentais" do calculador; interessa apenas que sejam em número finito. Aqui o argumento é muito simples e mesmo intuitivo: se o algoritmo dependesse de um número infinito de estados mentais, estes seriam confundidos uns com os outros e, portanto, não seria possível executar o cálculo com precisão.

Assim sendo, na formalização da máquina de Turing faz-se uso de um conjunto finito de estados Q. Os elementos de Q são os *estados da máquina*, que correspondem aos *estados mentais* do calculador. Exige-se também que, num dado instante, a cabeça de leitura de uma máquina de Turing só pode observar uma quadrícula da fita. Note-se que esta exigência é mais restritiva do que aquela que utilizamos na nossa análise do algoritmo da soma. Com efeito, no algoritmo da soma são observados três quadrículas de cada vez, e há dois estados mentais distintos, que designámos por S e T. Mas é possível imaginar que, em vez de observar as três quadrículas de uma só vez, como foi feito, podemos começar por observar o dígito de a, memorizando-o, depois passar ao dígito de b, memorizando-o também, e depois escrever o dígito

apropriado no espaço correspondente. Isto aumenta o número de estados mentais (já que o estado mental *guarda* os dígitos memorizados). Mas de facto, se o número de quadrículas observadas num só instante for finito, é sempre possível fazer uma transformação semelhante para o caso em que só uma quadrícula é observada de cada vez.

22. Chegado a este ponto, Turing começa a dissecar as acções executadas num cálculo genérico: escrita de símbolos e mudança do foco de atenção.

> *«Vamos imaginar que as operações executadas pelo calculador são divididas em "operações simples," que são tão elementares que não é fácil imaginá-las mais divididas. Uma tal operação consiste numa mudança no sistema físico formado pelo calculador e pela sua fita quadriculada. Conhecemos o estado deste sistema quando sabemos qual a sequência de símbolos na fita, quais destes são observados pelo calculador (possivelmente numa ordem particular), e qual o estado mental do calculador. Podemos assumir que uma operação simples modifica não mais que um símbolo. Quaisquer outras modificações podem ser decompostas em modificações simples deste tipo. No que diz respeito às quadrículas cujos símbolos podem ser alterados desta forma, a situação é a mesma que no caso das quadrículas observadas. Portanto, podemos assumir sem perda de generalidade que as quadrículas que podem ser modificadas são sempre uma parte das quadrículas que estão a ser "observadas".*
>
> *Para além destas modificações de símbolos, as operações simples devem incluir mudanças na distribuição das quadrículas observadas. As novas quadrículas observadas devem ser imediatamente reconhecíveis pelo calculador. Penso que é razoável pressupor que só devem ser quadrículas cuja distância às quadrículas observadas no instante imediatamente anterior não exceda um certo número fixo. Digamos que qualquer nova quadrícula observada tem, entre si e uma quadrícula observada no instante anterior, uma distância máxima de L quadrículas.*

Como na nossa formalização a cabeça de leitura só observa um quadrado da fita, então só pode escrever um símbolo a cada instante da computação; como a fita é unidimensional, a cabeça de leitura só se pode deslocar para a esquerda ou para a direita. A nossa restrição é tal que só permite que a cabeça de leitura se desloque do quadrado onde se encontra para um quadrado imediatamente adjacente. De novo, esta restrição é de pouca

importância; se for necessário mover a cabeça de leitura L quadrados numa certa direcção, esta pode ser deslocada um quadrado de cada vez, codificando no estado da máquina um número de 1 a L.

No algoritmo da soma as nossas acções foram: (1) escrever um certo dígito na quadrícula debaixo, e (2) passar a observar as três quadrículas imediatamente à esquerda do nosso foco de atenção.

23. Em seguida, Turing conclui a sua análise: as operações feitas dependem dos símbolos observados e do estado mental, e consistem na escrita de símbolos e na mudança do foco e do estado mental. No caso do algoritmo da soma apresentado acima, esta dependência é explicitada pelas frases "Se estamos no estado mental S (...)" e "O nosso objectivo no estado mental T (...)", junto com as tabelas correspondentes.

[p. 251] *«As operações simples devem portanto incluir:*

(a) Mudança do símbolo nalguma das quadrículas observadas.

(b) Mudança de uma das quadrículas observadas para outra quadrícula que esteja a menos de L quadrículas de distância de alguma das quadrículas anteriormente observadas.

É possível que algumas destas operações envolvam necessariamente uma mudança no estado mental. Portanto, uma operação singular, no caso mais geral, será uma das seguintes:

(A) Uma possível mudança (a) de símbolo, acompanhada de uma possível mudança de estado mental.

(B) Uma possível mudança (b) das quadrículas observadas, acompanhada de uma possível mudança de estado mental.

A operação a executar é determinada, como foi sugerido na página 250, pelo estado mental do calculador e os símbolos observados. Em particular, eles determinam o estado mental do calculador depois da operação ser executada.

Podemos agora construir uma máquina que faça o trabalho deste calculador. A cada estado mental fazemos corresponder uma m-configuração da máquina[16]*. A máquina lê B quadrículas que correspondem às B quadrículas observadas pelo calculador. Num só movimento a máquina pode modificar um símbolo numa quadrícula lida ou pode mudar qualquer das quadrículas lidas para outra quadrícula a uma distância não superior a L quadrículas de alguma outra quadrícula observada. O movimento a* [p. 252]

[16]N.T. No original o estado interno da máquina é chamado de «m-configuração».

> *executar, e a configuração que se segue, são determinados pelo símbolo lido e a m-configuração.»*

Formalmente, utilizamos uma função δ, chamada *função de transição*, para especificar as operações feitas. Se a operação a executar é determinada pelo estado da máquina e pelo símbolo observado, e se é necessário especificar o símbolo escrito, a mudança de foco e do estado mental, então a função terá a seguinte assinatura: $\delta : Q \times \Sigma \to Q \times \Sigma \times \{E, D\}$. Ou seja, para cada par (q, σ), em que $q \in Q$ é um estado da máquina, e $\sigma \in \Sigma$ é um símbolo, a função δ especifica

$$\delta(q, \sigma) = (q', \sigma', M),$$

em que $q' \in Q$ é um estado da máquina, $\sigma' \in \Sigma$ é um símbolo e $M \in \{E, D\}$ é uma direcção (esquerda ou direita).

A igualdade $\delta(q, \sigma) = (q', \sigma', M)$ significa: *se a máquina estiver no estado q e a cabeça de leitura ler o símbolo σ, então a máquina vai transitar para o estado q', e a cabeça de leitura vai substituir o símbolo σ pelo símbolo σ', e vai deslocar-se para a direcção M.* Ou seja, δ especifica o comportamento da máquina em termos das operações elementares que surgiram na análise de Turing.

E assim terminamos este texto, fazendo votos que a análise de Turing aqui traduzida tenha convencido o leitor que *toda a função que pode ser calculada por um procedimento finito pode também ser computada por uma máquina de Turing.*

Agradecimentos

Estou muito grato à Isabel Serra, ao Sérgio Loff e ao João Loff Barreto pela correcção atenta do meu Português, desvirtuado por vários anos a ler e falar quase exclusivamente em Inglês. Agradeço também ao José Félix Costa pelas várias conversas e investigações conjuntas sobre computabilidade. Por fim, estou grato ao árbitro do paper pela sua leitura atenta e minuciosa, que resultou, em particular, numa tradução mais fiel ao original.

Referências

[1] William Aspray, Charles Gillispie, Frederik Nebeker, and Albert Tucker. The Princeton mathematics community in the 1930s. Published online at `http://www.princeton.edu/mudd/math/`, 1985.

[2] Alonzo Church. An unsolvable problem of elementary number theory. In *Bulletin of the American Mathematical Society*, volume 41, pages 332–333, 1935.

[3] Alonzo Church. An unsolvable problem in elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[4] Jack Copeland and Diane Proudfoot. Turing's neural networks of 1948. Available online at `http://www.alanturing.net`, 2000.

[5] Martin Davis. Hilbert's tenth problem is unsolvable. *The American Mathematical Monthly*, 80(3):233–269, 1973.

[6] Martin Davis. *Engines of Logic*. W. W. Norton & Co., 2001.

[7] Nachum Dershowitz and Yuri Gurevich. A natural axiomatization of computability and proof of Church's thesis. *The Bulletin of Symbolic Logic*, 14(3):299–350, 2008.

[8] Robin Gandy. Church's thesis and principles for mechanisms. In Jon Barwise, H. Jerome Keisler, and Kenneth Kunen, editors, *The Kleene Symposium*, pages 55–111. North-Holland, 1980.

[9] Robin Gandy. The confluence of ideas in 1936. In Rolf Herken, editor, *The universal Turing machine: a half-century survey*, pages 55–111. Oxford University Press, 1988.

[10] David Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 9(10):437–479, 1902.

[11] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. Springer–Verlag, Berlin, 1928. Translated edition: Principles of Theoretical Logic (R.E. Luce, translator and editor), AMS Chelsea Publishing, New York, 1950.

[12] Philip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. Oxford University Press, 2006.

[13] Stephen Kleene. General recursive functions of natural numbers. *Mathematische Annalen*, 112(1):727–742, 1936.

[14] Hendrik W. Lenstra, Jr. Solving the Pell equation. *Notices of the American Mathematical Society*, 49(2):182–192, 2002.

[15] Bruno Loff. Physics, computation and definability. Master's thesis, Instituto Superior Tecnico, 2007.

[16] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. A proposal for the Dartmouth summer research project on artificial intelligence, 1955. Reprinted online at `http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html`.

[17] Toby Ord. Hypercomputation: computing more than the Turing machine. Honor's thesis, University of Melbourne, 2002.

[18] Emil Post. Finite combinatorial processes — formulation I. *Journal of Symbolic Logic*, 1(3):103–105, 1936.

[19] John Barkley Rosser. Highlights of the history of the lambda–calculus. *IEEE Annals of the History of Computing*, 6(4):337–349, 1984.

[20] Wilfried Sieg. Step by recursive step: Church's analysis of effective calculability. *The Bulletin of Symbolic Logic*, 3(2):154–180, 1997.

[21] Wilfried Sieg. Only two letters: The correspondence between Herbrand and Gödel. Technical report, Carnegie Mellon University, Department of Philosophy, 2004.

[22] Wilfried Sieg. Church without dogma — axioms for computability. In S. B. Cooper, B. Löwe, and A. Sorbi, editors, *New computational paradigms: Changing conceptions of what is computabl*, pages 139–152. Springer–Verlag, Berlin, 2008.

[23] Peter Smith. *An Introduction to Gödel's Theorems*. Cambridge University Press, 2007.

[24] Alan Turing. On computable numbers, with an aplication to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, 42(1):230–265, 1936.

[25] Alan Turing. Computability and λ-definability. *Journal of Symbolic Logic*, 2(4):153–163, 1937.

[26] Alan Turing. Intelligent machinery. Technical report, National Physical Laboratory, London, 1948. Available online at `http://www.alanturing.net/turing_archive/archive/l/l32/L32-001.html`.

[27] Alan Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.