2. O problema da decisão e a máquina universal de Turing[1]

Fernando Ferreira

Em Setembro de 1928, realizou-se em Bolonha o VI Congresso Internacional de Matemáticos. Foi o primeiro congresso em que participou

1 O editor deste volume pediu-me para escrever um texto para um público não especialista. Os meus instintos de profissional impeliram-me a querer dar referências, a elaborar, a abordar pontos subtis, a não condescender com formulações mais apelativas mas menos corretas. Mas, o que é facto, é que aceitei o pedido. Acabei por escrever um texto escorrido, quase sem elaborações e sem notas de pé de página. Juntei dois apêndices com informação adicional que não quis que figurasse no texto principal. Observações e referências (que tentei que fossem em português) foram colocados num anexo denso intitulado "Apontamentos complementares".

a Alemanha depois da primeira guerra mundial. À frente da delegação alemã (a segunda maior, depois da italiana) figurava David Hilbert, talvez o maior matemático ao tempo, professor na Universidade de Göttingen e já perto da reforma. Trabalhou e obteve resultados fundamentais e de grande influência em álgebra, teoria dos números, geometria, análise e física teórica, mas também se dedicou à lógica matemática e aos fundamentos de matemática. Na sua comunicação ao congresso, fez um apelo:

> " É claro que, para o completo desenvolvimento desta tarefa, é necessária a fervorosa colaboração da geração mais jovem de matemáticos."

De que tarefa falava Hilbert? Tratava-se, sem mais nem menos, de pôr a matemática em solo absolutamente seguro. Sem dúvida que o leitor estar-se-á a questionar: – Mas não é a matemática uma ciência completamente rigorosa? Não é a matemática absolutamente segura?

É pouco conhecido entre o público em geral que a matemática sofreu uma grande transformação na viragem do século XIX para o século XX, apenas comparável ao aparecimento da axiomática na Grécia antiga ou à descoberta do cálculo infinitesimal por Leibniz e Newton. A transformação consistiu essencialmente na aceitação sem pejo do infinito atual (o infinito tomado como acabado e não em potência, como quando se toma a sucessão de todos os números naturais 1, 2, 3, … como algo fechado e completo e não como um processo sem fim) e no abandono, por completo, da intuição como método de *justificação* matemática. Foi, também, acompanhada pela aceitação de abstrações cada vez maiores e cada vez mais afastadas das fontes da física e das aplicações. Podemos dar como exemplo os trabalhos dos matemáticos Richard Dedekind e Karl Weierstrass na fundamentação rigorosa do cálculo infinitesimal e integral. Estes trabalhos puseram em evidência a necessidade do infinito atual no tratamento rigoroso dos números reais. Um número real (um ponto do contínuo da reta) é agora visto como um ente de natureza infinitária (um número real é dado por uma dízima *infinita*). Um ponto é um ente infinitário: note o leitor o quão longe estamos da conceção euclidiana, onde um ponto é aquilo que não tem partes! Por sua vez, a criação por Georg Cantor de uma teoria extremamente bela e fecunda do infinito atual, em que se distinguem

vários tipos de infinitos – a teoria dos conjuntos – leva o tratamento do infinito atual até às suas últimas consequências.

O infinito atual tem sido objeto de desconfiança por parte de filósofos e matemáticos. Em 1902 cai sobre a comunidade matemática uma espécie de bomba: o paradoxo de Russell. Dada uma qualquer condição (p. ex., a condição de ser homem) podemos considerar a sua extensão (o conjunto de todos os homens). Normalmente, um conjunto não é membro de si próprio. O conjunto de todos os homens não é um homem. Mas, por vezes, é-o: o conjunto de todas as abstrações é, ele próprio, uma abstração. Bertrand Russell considerou o conjunto de todos os conjuntos que não são membros de si próprios. Põe-se a questão: é o conjunto de Russell membro de si próprio? Se é, então dada a sua definição (trata-se do conjunto formado pelos conjuntos que *não são* membros de si próprios), não é. Mas se não é, é. Chegamos a uma situação paradoxal: o conjunto de Russell nem pode ser membro de si próprio, nem pode deixar de o ser!

A descoberta do paradoxo de Russell provocou reações díspares e, por vezes, emotivas, entre os matemáticos. Uns, os tradicionalistas, que se opunham à emergente teoria dos conjuntos, viram na antinomia de Russell a prova de que o novo caminho era um erro. Outros, os modernos, atarefaram-se a tentar compreender as razões para o aparecimento do paradoxo e propuseram programas de fundamentação da matemática. Hilbert encontrava-se entre estes últimos e, nos anos vinte do século passado, propôs o seu próprio programa de fundamentação da matemática: o formalismo. Este programa competia com o programa logicista do próprio Russell e com o denominado programa intuicionista do matemático holandês L. E. J. Brouwer.

O programa de Hilbert concebia a matemática como uma espécie de jogo. No jogo de xadrez há a partida propriamente dita, jogada de acordo com regras exatas. Há, também, a teoria *sobre* o jogo de xadrez. Faz parte desta teoria a afirmação de que não é possível forçar xeque-mate com apenas dois cavalos a um rei isolado. Já não se trata agora de jogar uma partida de xadrez mas sim de investigar o jogo em si. É *metaxadrez*. A matemática, quando vista formalmente, também tem regras bem definidas. Elas são dadas pelos axiomas e pelas regras de inferência. Os axiomas correspondem ao tabuleiro inicial duma partida

de xadrez – as peças na sua posição inicial – e as regras de inferência correspondem às jogadas permitidas. A ideia fundamental do formalismo é ver a matemática como um jogo (dedutivo) simbólico. As peças deste jogo simbólico são as fórmulas da linguagem da matemática. Para que este jogo tenha regras exatas, como acontece no xadrez, é necessário simbolizar (formalizar) completamente a linguagem da matemática. Não basta enunciar o primeiro axioma da geometria euclidiana dizendo que por dois pontos passa uma reta. É necessário formulá-lo numa linguagem exata. Na notação da lógica simbólica, o axioma de Euclides escreve-se assim:

$$\forall x \forall y (Px \wedge Py \rightarrow \exists z (Rz \wedge Ixz \wedge Iyz))$$

Se o leitor não tem treino lógico e não percebe a fórmula acima tem, do ponto de vista estritamente formalista, uma vantagem sobre mim. Não consegue interpretar a fórmula acima. Mas esse é precisamente o ponto do formalismo: a fórmula deve ser vista como não querendo dizer nada, como sendo uma mera sequência de símbolos. Se o leitor tiver treino lógico adivinha que se pode interpretar os objetos de tipo *P* como pontos, os objetos de tipo *R* como retas e a relação *I* como a relação de incidência entre um ponto e uma reta (diz-se que um ponto *incide* numa reta se o ponto está na reta). Mas, para efeitos do programa formalista da fundamentação da matemática, não o devemos. Os axiomas de Euclides são vistos apenas como fórmulas de uma dada linguagem formal. Estas fórmulas são construídas de forma precisa a partir de símbolos primitivos (*P*, *R* e *I*) e de operadores lógicos como *e*, *ou*, *não*, *se … então …*, *para todo* e *existe* (na simbologia da lógica, $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\forall$ e $\exists$, respetivamente).

Para além dos axiomas próprios da teoria em questão (no nosso exemplo, os axiomas de geometria euclidiana), o sistema formal inclui também axiomas comuns a todos os sistemas: os axiomas da lógica (mais precisamente, os axiomas do denominado cálculo de predicados). Por exemplo, as fórmulas da forma "se *A*, então *A*" (na simbologia da lógica, "$A \rightarrow A$") podem fazer parte duma axiomática da lógica. As regras de inferência permitem gerar (deduzir) novas fórmulas a partir de outras fórmulas. Uma regra de inferência básica é a regra do *modus ponens*. Por exemplo:

Se *x* incide sobre *z*, então *x* é um ponto

x incide sobre *z*

Logo, *x* é um ponto

Na notação da lógica, podemos inferir a configuração simbólica "*Px*" a partir das configurações "*Ixz* → *Px*" e "*Ixz*". O "jogo de fórmulas" consiste em gerar certas sequências de símbolos (os teoremas) a partir de determinadas sequências dadas inicialmente (os axiomas) por intermédio das regras de inferência. Hilbert escreve:

> (...) não devemos todavia perder de vista o pré-requisito essencial do nosso procedimento. Pois existe uma condição, uma só mas absolutamente necessária, a que está submetida a utilização do [nosso] método (...), e essa condição é a *demonstração de consistência*.

Hilbert exige que, no cálculo axiomático formal, não se chegue a contradições, i.e., não se chegue a fórmulas da forma "*A* e *não-A*" (simbolicamente "*A* ∧¬*A*"). O "jogo", neste caso, deixaria de ter interesse até porque, duma contradição, é possível inferir qualquer fórmula. Por isso o paradoxo de Russell é tão devastador: toda a fórmula seria um teorema.

Na sua palestra em Bolonha, Hilbert menciona vários problemas relacionados com o seu programa de fundamentação da matemática. Um é o *problema de consistência* que acabámos de discutir. Hilbert pretendia mostrar que as regras do jogo formal das axiomáticas importantes para a matemática – um jogo cujas regras são completamente exatas, como é o xadrez – nunca levariam à dedução de contradições. Deduzir teoremas a partir duma axiomática é fazer matemática mas mostrar que a axiomática da aritmética não leva a contradições é fazer *metamatemática*. A metamatemática ou teoria da demonstração foi a disciplina inventada por Hilbert para levar a cabo o seu programa. Outro dos problemas mencionados na palestra foi o *problema da completude*. No caso da axiomática da aritmética, este problema consiste em mostrar que, para qualquer asserção simbólica da linguagem formal da aritmética, ou ela ou a sua negação é um teorema. Um terceiro problema foi também mencionado em 1928, mas num livro de Hilbert com o seu aluno Wilhelm Ackermann. É o *problema da decisão*, também conhecido

pelo seu nome alemão: *Entscheidungsproblem*. Hilbert e Ackermann escreveram que "*o Entscheidungsproblem deve ser considerado o problema principal da lógica matemática*" (em itálico no original).

O problema da decisão é o problema de encontrar um método *efetivo* (também se diz *mecânico* ou *algorítmico*) de acordo com o qual, dada uma fórmula da linguagem do cálculo de predicados, se determina se essa fórmula é, ou não, um teorema da lógica (i.e. deduzível apenas a partir dos axiomas do cálculo de predicados). Um método ou procedimento é *efetivo* se:

1. puder ser descrito através dum número finito de instruções exatas;
2. produzir o resultado desejável ao fim dum número finito de passos (desde que se sigam as instruções sem erro);
3. puder, em princípio, ser executado por um ser humano apenas com a ajuda de papel e lápis;
4. não exigir nem criatividade nem perspicácia por parte do ser humano.

Os algoritmos que as crianças aprendem para efetuar as operações básicas da aritmética são exemplos de procedimentos efetivos. Para quem sabe um pouco de lógica, o método das tabelas de verdade para decidir se uma fórmula do cálculo proposicional é uma tautologia é um procedimento efetivo.

Hilbert e Ackermann propuseram o seguinte problema: encontrar um método efetivo para separar os teoremas da lógica das outras fórmulas. Para teorias importantes da matemática, uma solução positiva para o *Entscheidungsproblem* permitiria decidir, de modo efetivo, se uma fórmula é um teorema dessa teoria.

Uma solução positiva para os três problemas hilbertianos da completude, consistência e decisão subscreveria uma visão magnífica da matemática. A matemática poderia ser vista como um grandioso cálculo formal – consistente, completo e decidível. A visão leibniziana dum *calculus ratiocinator* no domínio da matemática teria sido plenamente justificada. Perante um problema matemático, por mais difícil que fosse, bastaria "pegar na pena e sentar-se ao ábaco e (na presença de um amigo, se se quiser) dizer um para o outro: *calculemus*". O cálculo seria

efetivo, cego, como na multiplicação em numeração decimal, e daria sempre resposta. A matemática seria segura (consistência) e responderia a todas as questões (completude). Não só não haveria problemas insolúveis em matemática como as suas soluções seriam encontradas de modo efetivo. Numa comunicação em Königsberg (hoje a cidade russa de Kaliningrado) em 1931, por ocasião dum grande encontro de cientistas e médicos e da sua agraciação como cidadão honorário, Hilbert profere uma vez mais a sua fé inabalável no poder da razão humana com as seguintes palavras: *Wir müssen wissen, wir werden wissen* (temos de saber, havemos de saber). Estas palavras estão gravadas no seu túmulo em Göttingen. Era esta a genial visão hilbertiana da matemática. O apelo de Hilbert à nova geração não era um apelo qualquer. Era um apelo cheio de ambição, um apelo à resposta a questões de natureza fundamental. Era um apelo à resolução definitiva dos problemas dos fundamentos da matemática e à "*honra do próprio conhecimento humano*".

Não se tinham passado ainda três anos sobre a palestra de Hilbert em Bolonha quando o jovem matemático austríaco Kurt Gödel (nascido em Brno, hoje na República Checa, em 1906) mostra que a axiomática da aritmética, desde que seja consistente, não é completa. Umas semanas mais tarde, mostra que não é possível demonstrar a consistência da aritmética por meio de métodos aceitáveis para Hilbert (os métodos da metamatemática). Curiosamente, Gödel fez o anúncio público do primeiro destes resultados também em Königsberg, precisamente na véspera da comunicação de Hilbert de 1931. A *volte-face* foi totalmente inesperada e apanhou de surpresa o mundo matemático. Os resultados de Gödel demoraram algum tempo a ser plenamente apreciados e compreendidos. Afinal, o programa de Hilbert não era possível de levar a cabo. Afinal o grande matemático estava errado.

M. H. A. (Max) Newman ensinava em Cambridge na altura do congresso de Bolonha. Esteve presente no congresso e certamente terá ouvido o apelo de Hilbert. Newman não era lógico, era um topologista. A topologia era, então, uma disciplina nova que estava a unificar e a generalizar várias partes da matemática e em cuja base a teoria dos conjuntos desempenhava um papel importante. Não é de admirar que Max Newman também se interessasse pelos fundamentos da matemática. Em Cambridge, a lógica e os fundamentos da matemática não eram propriamente uma novidade pois foi uma universidade pioneira

da lógica moderna. Russell e o seu co-autor Alfred North Whitehead foram *fellows* no Trinity College de Cambridge. O matemático e filósofo Frank Ramsey (prematuramente falecido, com um pouco menos de vinte e sete anos, em 1930) também foi *fellow* em Cambrige (do King's College) e o célebre aluno de Russell, o filósofo Ludwig Wittgenstein, ensinava em Cambridge nos anos trinta. Max Newman decidiu oferecer um curso de fundamentos da matemática, mas não ao estilo tradicional de Cambridge (i.e., incidindo sobre o programa russelliano do logicismo). Ensinou um curso ao estilo de Hilbert. A terceira parte do curso, lecionada na primavera de 1935, terminava com a exposição dos resultados de Gödel.

O jovem inglês Alan Mathison Turing, nascido em 1912, estudava em Cambridge desde 1931 e foi assistir à terceira parte do curso de fundamentos da matemática de Newman. Nesse curso foi exposto ao programa de Hilbert, aos teoremas de Gödel e ao *Entscheidungsproblem*. Como iremos descrever, Turing vai resolver o problema da decisão pela negativa. Numa entrevista nos anos setenta, Max Newman afirma:

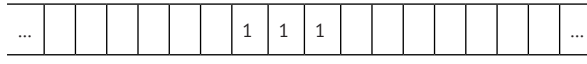
> Acredito que tudo tenha começado porque ele [Turing] frequentou um curso de fundamentos da matemática e de lógica dado por mim... Creio ter dito durante este curso que, por um processo construtivo, se entendia um processo puramente mecânico – e penso que terei mesmo dito que uma máquina o podia efetuar.

O leitor deve refletir no seguinte. A descrição de processo efetivo (ou mecânico) que mencionámos atrás é uma descrição um tanto imprecisa. Não é certamente uma definição matemática. Descreve, em traços largos, um conceito *fundamental*. Esta descrição é geralmente suficiente para sabermos que estamos perante um processo efetivo quando um deles nos é apresentado. Não é necessário saber, com rigor matemático, o que é um processo efetivo para nos convencermos, p. ex., que o algoritmo de multiplicação que aprendemos em criança é efetivo. Uma resposta negativa para o *Entscheidungsproblem* carece, porém, de uma abordagem diferente e Turing vai propor uma caracterização matematicamente precisa de processo efetivo. Newman acrescenta na citada entrevista:

> É claro que isto levou [Turing] a um novo desafio – que espécie de máquina – e isto inspirou-o a tentar resolvê-lo e a dizer o que seria uma máquina computacional perfeitamente geral.

O artigo de Turing de 1936 "On computable numbers with an application to the Entscheidungsproblem" (Sobre números computáveis com uma aplicação ao Entscheidungsproblem) é brilhante. Contém um análise conceptual do que é um processo efetivo e, concomitantemente, propõe uma definição matematicamente rigorosa do que se pode calcular com tais processos. Contém a descrição duma "super-máquina" com poder computacional para executar qualquer processo efetivo. Mostra que há problemas de decisão (i.e., de resposta "sim" ou "não") que não têm soluções através de processos efetivos. Finalmente, contém a solução para o *Entscheidungsproblem*.

O artigo introduz um simples e elegante dispositivo abstrato de natureza computacional, hoje conhecido por *máquina de Turing*. Trata-se de um dispositivo que opera de acordo com uma lista finita de instruções. O leitor deve ter em mente uma *imagem instantânea* da operação duma máquina de Turing como algo do género:



q Δ

Tem-se uma fita infinita (em ambas as direções), dividida em quadrados adjacentes (casas). Cada casa da fita pode ter impresso um *símbolo* ou está em branco. A máquina tem uma cabeça de leitura e escrita que, a cada momento, examina uma casa. Na figura acima, a cabeça da fita (Δ, na figura) está a examinar (ler) uma casa que tem o símbolo 1. Todas as casas à sua esquerda estão em branco. À direita da casa da cabeça estão dois símbolos 1 seguidos de casas em branco. A cabeça pode mover-se, de cada vez, uma casa para a esquerda ou para a direita. Finalmente, em cada momento, a (imagem instantânea da) máquina encontra-se num determinado estado discreto (há um número finito, pré-fixo, de estados, dependendo da máquina). No exemplo acima, a máquina encontra-se no estado q. A imagem instantânea da máquina pode ser representada por

q: … 0000000<u>1</u>110000000 …

onde os zeros representam casas em branco e o traço diz-nos onde se encontra a cabeça da máquina.

Uma máquina de Turing, propriamente dita, é uma lista finita de instruções, cada qual da seguinte forma: se a máquina está no estado E e a cabeça está a ler o símbolo X, então a máquina entra no estado F, substitui o símbolo X por Y e move a cabeça para a casa da esquerda (ou da direita). A lista não deve ser ambígua, i.e., não deve ter mais do que uma instrução aplicável quando a máquina está num determinado estado a ler um determinado símbolo. Por conveniência, permite-se também que Y possa ser o "símbolo branco", caso em que a instrução simplesmente apaga X. Como exemplo, considere-se a seguinte máquina (que denominamos M_1) com três instruções:

q	1	q	1	←
q	0	r	1	←
r	0	p	0	→

Acima, o símbolo 0 representa o "símbolo branco" e as setas são as instruções para mover a cabeça para a esquerda ou para a direita. As imagens instantâneas desta máquina podem encontrar-se num de três estados: q, r e p. A segunda instrução diz que se a (imagem instantânea da) máquina está no estado q com a cabeça a examinar uma casa em branco, então a máquina entra no estado r, imprime 1 e move-se uma casa para a esquerda. Se a configuração inicial da máquina é a imagem instantânea que discutimos acima, fica-se com a seguinte sequência:

q: … 0000000<u>1</u>110000000 …

q: … 000000<u>0</u>1110000000 …

r: … 00000<u>0</u>11110000000 …

p: … 0000000<u>1</u>1110000000 …

O leitor está a ver como é que a "máquina" funciona? Note que a máquina para na quarta configuração acima, pois não há nenhuma instrução que comece por "p1". O leitor deve convencer-se de que, perante uma

configuração inicial no estado q, com a cabeça sobre o símbolo mais à esquerda duma sequência finita de 1s (uns) consecutivos, o resultado da execução da lista de instruções é adicionar um 1 imediatamente à esquerda do primeiro 1, parando sobre a casa desse 1.

Vamos dar mais alguns exemplos de máquinas de Turing. Com alguma disciplina e concentração, estes exemplos não são difíceis de seguir. O leitor menos interessado pode ignorar a análise dos exemplos sem deixar de compreender o essencial mas, a meu ver, perde a oportunidade de adquirir "sensibilidade" sobre o que é um processo efetivo, sobre o que é um *programa*. Considere-se a máquina M_2:

q	1	r	1	→
r	1	q	1	→
q	0	q	0	→
r	0	p	0	←

Quando M_2 começa no estado q, com a cabeça sobre o símbolo mais à esquerda duma sequência finita (palavra) de 1s consecutivos, para se o número de símbolos é ímpar. Caso contrário, a cabeça prossegue para a direita sem nunca parar. O leitor beneficiará se experimentar alguns exemplos simples para se convencer de que M_2 opera como se diz (experimente com 111 e 1111).

Uma pequena modificação desta máquina, com dois estados especiais s (para "sim") e n (para "não"), decide se o número de 1s é par ou não, parando no estado correspondente. É a máquina M_3:

q	1	r	1	→
r	1	q	1	→
q	0	s	0	→
r	0	n	0	←

Estas três máquinas são bastante simples. No Apêndice I damos a lista de instruções duma quarta máquina M_4, um pouco mais complicada. É uma máquina que duplica o número de 1s quando se inicia no estado q com a cabeça no símbolo mais à esquerda duma sequência consecutiva de 1s.

Nesta altura, é conveniente introduzir alguma terminologia. Dada uma máquina de Turing M e uma palavra α (a *entrada*), considere-se a configuração inicial que está no estado q (o estado no canto superior esquerdo da lista de instruções, designado por *estado inicial*) e com a cabeça a examinar o símbolo mais à esquerda de α (o resto da fita está em branco). Se a máquina parar, escreve-se M(α)↓. Se não parar, escreve-se M(α)↑. P. ex., $M_2(111)\downarrow$ e $M_2(1111)\uparrow$.

Uma *máquina de decisão* é uma máquina de Turing M que para sempre (seja qual for a palavra de entrada) e que, quando para, fica num dos estados especiais s (resposta afirmativa) ou n (resposta negativa). A máquina M_3, considerada acima, é uma máquina de decisão. Para no estado s se a entrada tem um número par de 1s e para no estado n no caso contrário.

Dada uma máquina M e uma entrada α, se a máquina parar com a cabeça numa casa que não está em branco, a *saída* é a palavra que se inicia nessa casa e se prolonga para a direita até encontrar a primeira casa em branco. Se λ é a saída, escrevemos M(α) = λ. A primeira máquina considerada, a máquina M_1, adiciona sempre uma unidade à entrada. P. ex., $M_1(111) = 1111$. Em geral, $M_1(\alpha) = \alpha+1$. A máquina M_4 "duplica" a entrada: $M_4(\alpha) = 2\alpha$.

Não há dificuldade conceptual em estender as definições anteriores a máquinas que aceitam duas ou mais entradas. As entradas podem ser dadas separando-as por uma casa em branco. A seguinte máquina, denominada de M_5, efetua a soma de dois inteiros positivos (dados como sequências finitas de 1s). P. ex., quando a configuração inicial da fita é

q: … 0000<u>1</u>1101111111000000 …

(entradas 3 e 7) a máquina para na configuração final

p: … 0000<u>1</u>1111111110000000 …

(a saída é 10). O programa de M_5 é:

q	1	q	1	→
q	0	r	1	→
r	1	r	1	→
r	0	u	0	←
u	1	v	0	←
v	1	v	1	←
v	0	p	0	→

Este programa faz o seguinte: preenche com o símbolo 1 o espaço em branco que separa as duas entradas, apaga o último símbolo 1 da segunda entrada e, finalmente, coloca a cabeça no símbolo 1 mais à esquerda da fita. Escrevemos $M_5(111,1111111) = 1111111111$. Em geral, $M_5(\alpha,\beta) = \alpha+\beta$.

A grande ideia de Turing foi ver que as máquinas podem, *elas próprias*, ser consideradas entradas para máquinas de Turing. A máquina M_1, com três instruções, é descrita pela palavra q1q1←q∗r1←r∗p∗→ (para se evitar possíveis ambiguidades, denota-se o símbolo em branco das instruções da máquina em causa – aqui M_1 – pelo símbolo∗). Nada impede que se considere a configuração

…			q	1	q	1	←	q	*	r	1	←	r	*	p	*	→		…

E Δ

e que se tenha uma lista de instruções num alfabeto (finito) que contenha os símbolos q, r, p, ∗, 1, → e ← e estados E e outros. Mais interessantemente, podemos considerar uma configuração inicial com duas entradas, a primeira das quais uma lista de instruções duma máquina de Turing e a segunda uma "entrada" para a máquina cuja lista de instruções consta da primeira entrada. No caso da máquina M_1 e da entrada 111, ficar-se-ia com a seguinte configuração inicial:

… 00000q1q1←q∗r1←r∗p∗→01110000 …

Turing imagina agora uma lista de instruções que permita *seguir listas de instruções*. Uma máquina U programada assim (uma máquina *universal*), com entradas α_M e β – onde α_M é a lista de instruções de uma

dada máquina de Turing M – tem como saída (quando há saída, quando M(β)↓) a palavra M(β). Intuitivamente, a máquina U opera sobre β seguindo as instruções da lista α_M. A lista de instruções de U está concebida de tal modo que isto aconteça. Temos, por exemplo, $U(\alpha_{M1},\beta) = \beta+1$ e $U(\alpha_{M4},\beta) = 2\beta$. Em geral, temos a equação:

$$U(\alpha_M,\beta) = M(\beta).$$

Por exemplo:

$$U(q1q1{\leftarrow}q{*}r1{\leftarrow}r{*}p{*}{\rightarrow},111) = 1111.$$

Na secção 7 do seu artigo, Turing esboça em pouco mais de três páginas como construir uma lista de instruções duma máquina universal. Como o leitor imaginará, não é simples dar uma descrição detalhada duma máquina universal de Turing dado que é uma tarefa muito propensa a "erros de programação". Quem já programou sabe perfeitamente que é comum cometer lapsos e até erros de conceção, especialmente se a linguagem de programação for muito pouco flexível (como é o caso das listas de instruções das máquinas de Turing). Na programação duma máquina universal, a procura e execução da instrução relevante da máquina de entrada é um processo cheio de passos artificiosos. Também é muito artificioso alocar memória (apesar de haver todo o espaço do mundo, visto que a fita é infinita).

O artigo de 1936 contém erros de programação que foram parcialmente emendados pelo próprio Turing numa correção publicada um ano depois. Mas lapsos e erros subsistiram e um colaborador de Turing (Donald W. Davies) apontou-lhe em 1947 alguns erros graves de programação:

> Quando disse isto a Turing, ele ficou impaciente e disse-me claramente que estava a perder o meu tempo e o dele com esforços sem valor.

Bastante mais tarde, numa entrevista nos anos setenta, Davies reporta que Turing "ficou muito irritado e (lhe) disse furiosamente que a questão realmente não interessava, que a coisa estava correta em princípio". Turing tinha razão: do ponto de vista conceptual estes detalhes não interessam e a ideia de máquina universal é correta. Em 1947, o trabalho

já era conhecido pelos especialistas e fora imediatamente aceite como uma obra fundamental. Porém, os detalhes já interessariam a quem quisesse implementar, na prática, uma máquina universal. Se bem que as máquinas de Turing sejam teoricamente muito claras, não são adequadas do ponto de vista prático (p. ex., no que diz respeito a cálculos importantes que devem ser efetuados tão rapidamente quanto o *hardware* o permita) por causa duma série de problemas, alguns já referidos, como os de alocação de memória e sua consulta trabalhosa, alocação de espaço para a computação propriamente dita, grande ineficiência, necessidade de manobras muito artificiosas, etc. Tudo isso torna a programação muito difícil e pouco transparente, cheia de detalhes incidentais e de difícil articulação. Um código de programação ao nível da máquina (i.e., que especifica as operações básicas que a máquina faz) para ser *prático* deve utilizar operações básicas que possam ser realizadas de modo eficiente e confiável por meios eletrónicos e, além disso, ser relativamente fácil de utilizar por humanos. É, porém, crucial que o código seja suficiente para, em princípio (se não na prática, por limitações de memória e tempo), incorporar a ideia de máquina *universal*.

A história do aparecimento dos computadores eletrónicos digitais é fascinante mas este não é o lugar próprio para a contar. Não se deve ficar com a impressão de que Turing não se interessava por aplicações. Turing participou na história da construção de computadores, nomeadamente na construção de um dos primeiros computadores eletrónicos, o ACE (Automatic Computing Engine). O relatório técnico que Turing escreveu para o ACE (provavelmente nos últimos meses de 1945) continha discussões detalhadas de engenharia e, inclusivamente, avançava com uma verba concreta para o custo da construção da máquina. Naturalmente, a construção dum computador digital é em grande parte um empreendimento de engenharia. Porém, ainda que seja interessante falar dos problemas complicados de engenharia e de programação que tiveram que ser superados para construir os computadores eletrónicos assim como mencionar o incrível avanço tecnológico a que temos assistido desde meados dos anos quarenta, não se pode esquecer que venceu uma conceção de computador que desafia o *senso comum*. Ainda em 1956 (vinte anos depois do artigo seminal de Turing!), Howard Aiken, um dos pioneiros da computação e professor na Universidade de Harvard afirmava:

> … se porventura vier a acontecer que as bases lógicas de uma máquina concebida para a solução numérica de equações diferenciais coincida com as lógicas duma máquina concebida para a contabilidade de um armazém, considerá-la-ia a mais espantosa coincidência que alguma vez encontrei.

Este é o senso comum, mas os computadores atuais não são como o senso comum sugere. São computadores de *propósito geral*, capazes de ler *software* adequado, este sim de propósito específico. Os computadores atuais permitem ter como entradas instruções (*software*) desenhadas para desempenhar tarefas específicas: são, em suma, incorporações da ideia de máquina *universal*.

Hoje em dia os computadores são um utensílio doméstico, como uma televisão ou um frigorífico. Há qualquer coisa de pasmoso no facto de que, na raiz dos computadores pessoais que usamos hoje diariamente, esteja uma ideia (a ideia da *universalidade* em computação) que surgiu por intermédio de questões da fundamentação da matemática. Podemos fazer "história virtual" e imaginar um encadeamento de acontecimentos que tivesse levado à conceção e construção de computadores de propósito geral que não tivesse passado pelas discussões e problemas postos pela crise dos fundamentos da matemática do início do século XX. Mas o facto é que a "história real", a que realmente teve lugar, passou por questões extremamente teóricas (fundamentos da matemática). Não é este um caso único na história em que a ânsia pelo conhecimento e a vontade de compreender, assim como a procura desinteressada da verdade, estão na origem de grandes avanços tecnológicos.

Na parte final do seu artigo, como aplicação dos conceitos anteriormente introduzidos, Turing resolve o *Entscheidungsproblem*. O título da última secção é, precisamente, "Aplicação ao *Entscheidungsproblem*". Turing baseia-se em trabalho que efetuou numa secção prévia, onde exibe um problema de decisão que não pode ser decidido de modo efetivo (um problema destes diz-se *indecidível*). O problema é o seguinte: decidir, dadas entradas α_M e β, se a máquina M para com a entrada β, i.e., se $M(\beta)\downarrow$. A este problema chama-se o *problema da paragem* ("halting problem", em inglês). O raciocínio de Turing é por redução ao

absurdo. Turing vai supor que o problema da paragem é decidível e, a partir desta suposição, vai chegar a uma contradição.

Suponhamos então, com vista a um absurdo, que o problema da paragem é decidível. Então existe uma máquina de decisão H que decide este problema. Isto quer dizer que o seguinte vale sempre:

$H(\alpha_M,\beta)$ decide afirmativamente (para no estado s) se $M(\beta)\downarrow$, e

$H(\alpha_M,\beta)$ decide negativamente (para no estado n) se $M(\beta)\uparrow$.

Com a ajuda desta máquina H, podemos construir uma máquina *diagonal* D que, com a entrada α_M, procede do seguinte modo:

1. obtém o resultado da decisão $H(\alpha_M, \alpha_M)$;
2. se a decisão é afirmativa, executa para sempre a instrução de mover a cabeça para a direita (portanto, entra numa situação em que nunca para);
3. se a decisão é negativa, para imediatamente.

Por construção, esta máquina goza da seguinte propriedade:

$$D(\alpha_M)\downarrow \text{ se, e somente se, } M(\alpha_M)\uparrow,$$

qualquer que seja a máquina de Turing M. Ora, se a máquina M for a *própria* máquina D chega-se a uma contradição:

$$D(\alpha_D)\downarrow \text{ se, e somente se, } D(\alpha_D)\uparrow.$$

Por outras palavras, com a entrada α_D, a máquina D para se, e somente se, não parar! Isto é contraditório.

O problema da paragem é o problema indecidível paradigmático. Em geral, mostra-se que um dado problema é indecidível através da *redução* do problema da paragem a esse problema. A ideia é simples. Para mostrar que um problema é indecidível mostra-se que se fosse decidível então o problema da paragem também o seria. *Quod non* (o que não é o caso).

Turing reduz o problema da paragem ao *Entscheidungsproblem*, descrevendo um modo efetivo de associar a cada par M e β uma fórmula $F_{M,\beta}$ do cálculo de predicados de tal sorte que

$M(\beta)\downarrow$ se, e somente se, $F_{M,\beta}$ é um teorema do cálculo de predicados.

A decisão de paragem fica reduzida à decisão de ser teorema do cálculo de predicados. Logo, esta última não se pode fazer de modo efetivo. Com este argumento, Turing mostra que o *Entscheidungsproblem* é indecidível dando, portanto, uma resposta negativa à questão de Hilbert e Ackermann.

Quando Turing estava a terminar o seu artigo na primavera de 1936, tomou conhecimento de um trabalho do lógico americano Alonzo Church que também continha uma solução para o *Entscheidungsproblem*. O artigo intitulava-se "A note on the *Entscheidungsproblem*". Em ciência, é o autor que primeiro chega a um resultado que geralmente fica com o crédito de o ter obtido, mesmo que outro autor tenha independentemente chegado à mesma conclusão. Hoje a solução do *Entscheidungsproblem* é conhecida por *teorema de Church*. Apesar da publicação de Church, Max Newman apoiou e instigou a publicação do artigo de Turing já que o método deste era suficientemente diferente do de Church para merecer ser conhecido e disseminado. Isto é tanto mais verdade porque o artigo de Turing introduz o conceito de máquina universal e argumenta que a noção de máquina de Turing captura a noção informal de efetividade computacional. Na secção 9 do seu artigo, Turing observa que, até à data, não tinha havido nenhuma tentativa de mostrar que as caracterizações matemáticas de processos efetivos incluem, de facto, todos os processos que se podem naturalmente caracterizar de efetivos. É claro que um argumento que procure identificar uma noção matemática com uma noção informal (ou "natural") não pode ter o carácter duma demonstração matemática. No primeiro parágrafo da secção 9, Turing questiona-se: *"Quais são os processos possíveis que podem ser efetuados na computação (...)?"* A secção procura responder a esta pergunta através duma análise detalhada do que significa "um ser humano efetuar uma computação". A análise é necessariamente intuitiva e baseia-se num processo de eliminação e depuração em que detalhes irrelevantes para a computação são descartados e em que o processo computacional é reduzido a operações o mais simples possíveis. No

final, Turing chega à sua noção de máquina. A secção 9 do artigo de Turing pode ser considerada um dos mais bem sucedidos casos de *filosofia aplicada*.

A identificação da noção informal de função *efetivamente* computável com uma noção matematicamente rigorosa tinha sido primeiramente proposta, também por Church, num outro artigo do ano de 1936. Church escreve:

> O propósito do presente artigo é propor uma definição de computabilidade efetiva que corresponda de modo satisfatório à noção intuitivamente vaga em cujos termos os problemas desta classe [de problemas efetivos] são frequentemente formulados (...)

E, numa nota de pé de página, acrescenta que "a proposta para identificar estas noções com a noção de computabilidade efetiva é feita pela primeira vez neste artigo". As noções a que Church se refere são duas noções de computabilidade, uma devida a Church e ao seu estudante Stephen C. Kleene (em termos do denominado *cálculo lambda*) e a outra devida a Gödel e ao jovem matemático francês, prematuramente falecido com 23 anos, Jacques Herbrand (esta por meio da noção de *recursividade por equações*). Para efeitos da nossa discussão, não é importante saber o que são estas noções mas apenas que elas são matematicamente rigorosas e que a sua equivalência tinha sido estabelecida recentemente (mais tarde, no apêndice ao seu artigo de 1936, Turing mostra que a sua definição de computabilidade em termos de máquina de Turing coincide com a noção de Church e Kleene: as três noções são, pois, equivalentes). A identificação entre estas noções formais e a noção informal de computabilidade efetiva ficou conhecida na literatura por *tese de Church*. Estes assuntos foram objeto de troca de ideias por parte de Church, Gödel e Kleene. Porém, a identificação da noção informal com a noção de computabilidade por meio do cálculo lambda era "completamente insatisfatória" para Gödel ao tempo. Também, numa carta a Martin Davis em 1965, Gödel explica que, na altura, "não estava de todo convencido que o [meu] conceito de recursão compreendia todas as recursões possíveis". O que convenceu Gödel da correção da tese de Church foi o artigo de Turing. A propósito duma formulação geral dos seus teoremas da incompletude, Gödel escreve em 1965:

> A obra de Turing fornece uma análise do conceito de "processo mecânico" (ou antes "algoritmo" ou "processo de cálculo" ou "processo combinatório finito"). Mostra-se que este conceito é equivalente ao de uma "máquina de Turing".

Foi o exercício de filosofia aplicada de Turing que convenceu Gödel. O facto de ser possível dar uma definição *absoluta* da noção de computabilidade efetiva é classificado por Gödel como "uma espécie de milagre". Gödel contrasta a noção de computabilidade efetiva com a noção de *definibilidade*. Para esta noção já parece não haver milagres. A noção de definibilidade é *relativa* à linguagem onde se formula a definição. (Vale a pena analisar a situação com algum detalhe e o leitor pode encontrar uma discussão deste assunto no Apêndice II.)

No caso da computabilidade efetiva, dá-se um "milagre" porque é possível caracterizar os processos mecânicos básicos que dão origem a toda e qualquer computação efetiva, o que está estreitamente ligado a não ser possível decidir efetivamente se uma determinada lista de instruções básicas dá, ou não, origem a um processo que termina. Deixamos o leitor com as seguintes palavras de Gödel, numa comunicação ao bicentenário da Universidade de Princeton em 1946:

> Parece-me que esta importância [do conceito de computabilidade de Turing] é consideravelmente devida ao facto de que, pela primeira vez, se ter conseguido dar uma definição absoluta de uma noção epistemológica com interesse, i.e., absoluta no sentido de não depender do formalismo escolhido.

Apêndice I

A máquina M_4 é dada pela seguinte lista de onze instruções:

q	1	r	0	→
q	0	p	0	→
r	1	r	1	→
r	0	u	0	→
u	1	u	1	→

u	0	v	1	→
v	0	t	1	←
t	1	t	1	←
t	0	w	0	←
w	1	w	1	←
w	0	q	0	→

Esta máquina, quando é iniciada no estado q com a cabeça no símbolo mais à esquerda da sequência 111, tem os seguintes movimentos:

q: ... 000**1**110000000000 ...
r: ... 0000**1**10000000000 ...
r: ... 00001**1**0000000000 ...
r: ... 000011**0**000000000 ...
u: ... 0000110**0**00000000 ...
v: ... 00001101**0**0000000 ...
t: ... 0000110**1**10000000 ...
t: ... 000011**0**110000000 ...
w: ... 00001**1**0110000000 ...
w: ... 0000**1**10110000000 ...
w: ... 000**0**110110000000 ...
q: ... 0000**1**10110000000 ...
r: ... 00000**1**0110000000 ...
r: ... 000001**0**110000000 ...
u: ... 0000010**1**10000000 ...
u: ... 00000101**1**0000000 ...
u: ... 000001011**0**000000 ...
v: ... 0000010111**0**00000 ...
t: ... 0000010111**1**00000 ...
t: ... 00000101**1**1100000 ...
t: ... 0000010**1**111100000 ...
t: ... 000001**0**111100000 ...
w: ... 00000**1**0111100000 ...
w: ... 0000**0**10111100000 ...
q: ... 00000**1**0111100000 ...
r: ... 000000**0**111100000 ...
u: ... 0000000**1**11100000 ...
u: ... 00000001**1**1100000 ...

u:	... 00000001<u>1</u>100000 ...
u:	... 00000001<u>1</u>100000 ...
u:	... 00000001111<u>0</u>0000 ...
v:	... 00000001111<u>0</u>000 ...
t:	... 00000001111<u>1</u>1000 ...
t:	... 00000001<u>1</u>1111000 ...
t:	... 00000001<u>1</u>111000 ...
t:	... 00000001<u>1</u>1111000 ...
t:	... 00000001<u>1</u>11111000 ...
t:	... 00000000111111000 ...
w:	... 00000<u>0</u>0111111000 ...
q:	... 00000001111111000 ...
p:	... 00000001<u>1</u>11111000 ...

O leitor está a ver como é que esta máquina opera?

Apêndice II

No que se segue, vamos considerar a noção de definibilidade para sequências infinitas de zeros e uns (sucessão binárias). Um exemplo duma definição destas é: "a sequência que é 1 nas posições ímpares e é 0 nas posições pares". Esta definição caracteriza a sucessão

10101010101010101010101 ...

As definições são veiculadas por intermédio de frases duma dada linguagem. Ora, as frases definidoras podem ser enumeradas. Podemos, mesmo, *definir* uma tal enumeração. Tomemos, então, uma dessas enumerações

D1, D2, D3, D4, D5, D6, ...

onde cada Dn (com n = 1, 2, 3,...) é uma definição que dá origem a uma sucessão binária, designada por sn. Cada uma destas sucessões sn tem um número infinito de posições, cujos valores denotamos assim:

sn1, sn2, sn3, sn4, sn5, sn6, sn7, sn8, sn9,...

em que snk (com k = 1, 2, 3,...) denota o valor da k-ésima posição da sucessão sn. Note que cada snk ou é 0 ou é 1.

Agora podemos "diagonalizar" e *definir* a sucessão d ('d' de diagonalização) que é 0 nas posições k em que skk é 1, e que é 1 nas posições k em que skk é 0:

d1, d2, d3, d4, d5, d6, d7, d8, d9, …

Esta sucessão diagonal foi *definida*. Como as definições foram todas enumeradas, esta definição é, digamos, a i-ésima definição Di. Assim, a sucessão acima é

si1, si2, si3, si4, si5, si6, si7, si8, si9, …

ou seja, dk = sik para todo o k. Em particular, di = sii. Isto é um absurdo, pois d foi definida de tal modo que di é 0 se sii é 1 e di é 1 se sii é 0.

Qual é a conclusão a tirar? O que se passa é que não se pode pressupor que a definição da sucessão diagonal ainda possa ser feita na linguagem (formal) originariamente dada. Sempre que fixamos uma linguagem para fazer definições, há uma nova definição (a da sucessão diagonal) que *não pode* fazer parte da linguagem original. Por isso a noção de definibilidade é sempre relativa a uma linguagem.

No caso da computabilidade efetiva, os programas (listas de instruções) mantêm-se sempre na *mesma* linguagem. Este é o "milagre". Então, perguntará o leitor, o que é que impede que se utilize um argumento de diagonalização, à semelhança do caso da definibilidade, e se chegue a uma contradição? O que o impede é a indecidibilidade do problema da paragem pois essa indecidibilidade impossibilita a existência duma enumeração *efetiva* de todas as máquinas de Turing que calculam sucessões binárias infinitas.

Apontamentos complementares

A comunicação de Hilbert ao congresso de Bolonha encontra-se traduzida para português em (Hilbert 2003: 276-284) com o título "Problemas na fundamentação da matemática". O ensaio (Ferreira 1992) é um "divertimento" sobre paradoxos e, em particular, sobre o paradoxo de Russell. A enciclopédia organizada por João Branquinho e Desidério Murcho (Branquinho e Murcho 2001) contém várias entradas sobre temas mencionados no corpo deste artigo: paradoxo de Russell, logicismo,

formalismo, intuicionismo, programa de Hilbert, fundamentos da matemática, teoria dos conjuntos, máquina de Turing, funções recursivas, etc. As lutas dos fundamentos da matemática dos anos vinte da século passado não eram encaradas de ânimo leve pelos seus protagonistas, como se pode constatar pelo artigo (Ferreira 2008). Neste artigo também se descreve o programa de Brouwer. Trata-se de um programa especialmente interessante pois, se bem que tenha um cariz tradicionalista, é o mais revolucionário de todos.

A nossa descrição do programa de Hilbert está longe de fazer justiça à subtileza e sofisticação da posição hilbertiana. Roça, por vezes, a caricatura. Apenas descrevemos a faceta do programa que nos interessava para apresentar o *Entscheidungsproblem*. Insistimos na parte simbólica e formal, ignorando a componente finitista e o denominado método dos elementos ideais. Pode consultar-se (Ferreira 1995) para uma visão mais completa do programa de Hilbert. Os artigos de (Kahle 2006) e (Pereira 2006) também são informativos. A comparação do formalismo com o xadrez aparece num artigo de Hermann Weyl dos anos vinte. Esse artigo encontra-se traduzido para o inglês em (Mancosu 1998: 123-142) sob o título "The current epistemological situation in mathematics". A citação de Hilbert sobre a condição de consistência encontra-se no seu artigo "Sobre o infinito", traduzido para português em (Hilbert 2003: 234-255) assim como a alusão à honra do conhecimento humano. Esta alusão faz parte da seguinte frase (os destaques em itálico são do original): "queria deixar patente que a clarificação definitiva da *natureza do infinito* se tornou necessária, não apenas por interesse especial das diversas ciências particulares, mas antes para a *honra do próprio conhecimento humano*". O *Entscheidungsproblem* foi enunciado por Hilbert e Ackermann em termos semânticos: mostrar que existe um processo efetivo para determinar se uma fórmula do cálculo de predicados é uma verdade lógica (i.e., verdadeira sob todas as interpretações). Esta formulação é equivalente à do texto. A afirmação de que o *Entscheidungsproblem* é o principal problema da lógica matemática aparece em (Hilbert e Ackermann 1928: 77). Se é verdade que a visão hilbertiana da matemática repousava nas soluções positivas para os problemas da consistência e da completude, já se podem levantar dúvidas se também repousaria na solução positiva do *Entscheidungsproblem* pois Hilbert descreve este problema como um problema da lógica matemática e não como um problema dos fundamentos da matemática.

A locução *procedimento efetivo* vem duma terminologia estabelecida no inglês: "effective procedure". Neste artigo não estamos a utilizar a palavra 'efetivo' no seu sentido comum em português, mas sim num sentido técnico (discutido no texto). Os quatro pontos da caracterização de processo efetivo foram adaptados de (Copeland 2017). Uma solução positiva para o *Entscheidungsproblem* seria, não só magnífica, mas também aterradora para alguns. O célebre matemático inglês G. H. Hardy comenta em (Hardy 1929) que "se [o *Entscheidungsproblem* tivesse uma solução positiva] então teríamos um conjunto de regras mecânicas para a solução de todos os problemas da matemática e as nossas atividades - como matemáticos - terminariam". O estatuto dos matemáticos (e da matemática) ficaria bastante abalado já que o seu trabalho seria, em princípio, substituível por uma máquina. Acrescente-se que uma solução positiva para o problema da completude num determinado domínio da matemática (p. ex. a aritmética) teria como consequência a possibilidade de decidir mecanicamente se as asserções formais desse domínio são, ou não, teoremas. Esta observação (simples de justificar) parece ter escapado a Hilbert, tendo sido explicitamente feita por Turing na secção 11 do seu artigo de 1936.

A comunicação de Hilbert ao encontro de Königsberg encontra-se traduzida para o inglês em (Ewald 1996: 1157-1165) sob o título "Logic and the knowledge of nature". Os detalhes das vidas de Hilbert, Gödel e Turing mencionados no corpo do artigo podem encontrar-se em (Reid 1986), (Dawson 1997) e (Hodges 1992). Em português, vale a pena consultar (Goldstein 2009) que, apesar de alguma falta de rigor, é interessante. (Davis 2004) é uma muito boa obra de divulgação mas, infelizmente, a tradução para português é muito distraída. O resultado de Gödel sobre a incompletude da aritmética (o sistema que Gödel trata no seu artigo é um sistema de classes que contém a aritmética; NB, do ponto de vista matemático isso é irrelevante) tem uma hipótese mais forte do que a consistência (a denominada ω-consistência). Em 1936, J. B. Rosser mostra que o requisito da ω-consistência pode ser enfraquecido para a consistência. Os artigos de Gödel e Rosser encontram-se traduzidos para português em (Gödel et al. 2009). Os artigos originais não são o melhor lugar para quem se queira iniciar nestes assuntos. Já as lições de Gödel "Acerca de proposições indecidíveis de sistemas matemáticos formais" e a exposição de Rosser "Uma exposição informal das demonstrações dos teoremas de Gödel e de Church", ambas

em (Gödel et al. 2009), são bons pontos de partida. Há também manuais que explicam estes temas. Em português existem (Oliveira 2010) e, mais avançados, (Sernadas e Sernadas 2012) e (Carnielli e Epstein 2008). Este último contém discussões sobre questões dos fundamentos da matemática.

Os excertos das entrevistas de Newman foram citados a partir de (Copeland 2004: 206). Este livro, como o título indica, junta as principais publicações de Turing (incluindo, é claro, o "On computable numbers with an application to the Entscheidungsproblem" e a correção de 1937) e complementa-as com introduções e comentários. É claro que Turing não usou a expressão "máquinas de Turing", usou antes a expressão "a--máquinas" (para máquinas automáticas). A nossa descrição do artigo de Turing de 1936 é-lhe, em traços gerais, fiel mas modifica e simplifica (e omite) algumas discussões. Em primeiro lugar, as a-máquinas têm uma fita que é apenas infinita numa das direções tendo, portanto, um começo. As instruções para as a-máquinas são também ligeiramente diferentes daquelas que apresentámos e Turing adota convenções que não seguimos (como é o caso da divisão das casas em casas E, para *erasable*, e casas F, para *fixed*). Por outro lado, os números computáveis a que o título do artigo se refere são números reais, não são inteiros positivos. As listas de instruções para estes números têm como objetivo "imprimir" a sua representação decimal (infinita). Em conformidade com esta visão, a noção de paragem de Turing é diferente: é, essencialmente, a noção de entrar em círculo, caso em que a máquina fica presa na impressão duma casa decimal, não conseguindo prosseguir para a casa decimal seguinte. Turing usa a terminologia (pouco feliz) de máquinas "circulares" e máquinas "sem círculos". Ao problema de decidir se uma máquina não tem círculos, Turing chama-lhe o problema da satisfatoriedade. A expressão "halting problem" foi introduzida bastante mais tarde em (Davis 1958: 70). Turing apresenta duas demonstrações para a indecidibilidade. Sobre a primeira, Turing é duma grande candura. Escreve que *apesar da demonstração ser perfeitamente correta, tem a desvantagem de poder deixar no leitor a sensação de que "algo deve estar errado"* (destaque no original). Tanta simplicidade poderia deixar o leitor desconfiado… A demonstração que apresentamos no texto é uma versão da primeira demonstração de Turing. O leitor que queira saber *exatamente* o que Turing escreveu deve, é claro, ler o artigo original.

Há uma ocasião no nosso texto em que, para não quebrar a fluidez da exposição, comprometemos um pouco o rigor: foi na descrição das entradas de máquinas de Turing para a máquina universal. A máquina universal, sendo ela própria uma máquina de Turing, é dada por uma lista finita de instruções na qual constam apenas um número finito de estados e de símbolos. Por outro lado, as máquinas de Turing - como entradas para a máquina universal - consistem num número finito de estados e de símbolos que podem ser tão numerosos quanto se queira (o número finito depende da máquina). A máquina universal deve estar preparada para aceitar e trabalhar sobre entradas (listas de instruções) com um número tão grande quanto se queira de estados e símbolos. É fácil de tornear este problema por meio de codificações apropriadas. P. ex., podem-se codificar os estados q_1, q_1, q_3,... por EI, EII, EIII,... Bastariam dois símbolos (E e I) para descrever qualquer um deste número infinito de estados (não é necessário um símbolo novo para cada estado). *Mutatis mutandis* para a codificação de símbolos.

A citação de Davies foi extraída do terceiro parágrafo do artigo "Corrections to Turing's universal computing machine" publicado em (Copeland 2004). O comentário da entrevista de Davies é citado também a partir de (Copeland 2004). Esta obra, assim como o capítulo 8 de (Davis 2004), são bons lugares de consulta para quem estiver interessado na história da construção dos primeiros computadores eletrónicos de propósito geral. A citação de Aiken está no capítulo 7 do livro de Davis.

Os dois artigos de Church de 1936 podem encontrar-se na colectânea (Davis 1965). O segundo artigo que mencionámos, onde é formulada a tese de Church, intitula-se "An unsolvable problem of elementary number theory". A designação de "tese" para a proposta de Church é devida a Kleene num artigo de 1943. Veja-se (Davis 1965: 274) e (Davis 1982: 13). Por vezes, a tese de Church também é chamada de tese de Church-Turing. A classificação, por Gödel, da proposta de Church como sendo "completamente insatisfatória" aparece numa carta de 1935 de Church a Kleene, citada em (Davis 1982: 9). A carta de Gödel a Davis vem mencionada em (Gödel et al. 2009: 51-52), assim como os dois parágrafos de Gödel de 1965 (Gödel et al. 2009: 113). Gödel afirma que a noção de computabilidade é *absoluta*: o leitor deve comparar esta noção com a noção de definibilidade (discutida no Apêndice II) ou de dedutibilidade formal aritmética (não é absoluta pois depende da axiomática de

base). A locução "uma espécie de milagre" e a citação final também se encontram em (Gödel et al. 2009: 883). Estamos a referir-nos ao artigo "Reflexões sobre problemas em matemática apresentados à conferência do bicentenário de Princeton". Apesar do contraste que Gödel faz da noção de computabilidade com as noções de demonstrabilidade e definibilidade o artigo é, realmente, uma tentativa de salvar estas noções do seu relativismo à linguagem. A noção epistemológica a que Gödel se refere é, presumivelmente, o conhecimento que é obtido apenas por meio dum cálculo - o que é identificado com o conceito matemático de computabilidade à Turing.

Na internet encontra-se disponível bastante informação sobre Turing. Em http://www.turingarchive.org está o "The Turing Digital Archive". Há também a página http://www.turingcentenary.eu/ do "2012 The Alan Turing Year". É possível encontrar o artigo de Turing de 1936 na internet. Também é possível encontrar na internet os meus artigos mencionados nestes apontamentos.

Agradecimentos

Quero agradecer a leitura atenta e os comentários do José Carlos Espírito Santo, Gilda Ferreira, Sérgio Fernandes, José Almeida Santos, Reinhard Kahle, Luís Moniz Pereira, Augusto Franco de Oliveira e da Malu. As suas observações e críticas foram muito importantes para melhorar este trabalho e torná-lo mais acessível mas, é claro, a responsabilidade da versão final e das suas falhas é inteiramente minha. Quero também agradecer à Olga Pombo por me ter dado a conhecer a citação de Leibniz sobre o *calculemus* (Leibniz 1960: 200) e tê-la traduzido do latim para português. Finalmente, devo uma palavra de agradecimento ao amável convite do organizador deste volume e à sua louvável iniciativa de, por este meio, celebrar o centenário do nascimento de Alan Turing. Um muito obrigado ao José Carlos Espírito Santo.

A FCT - Fundação para a Ciência e Tecnologia (através dos projetos PTDC/FIL-FCI/109991/2009 e PEst-OE/MAT/UI0209/2013) e a Faculdade de Ciências da Universidade de Lisboa (por meio da concessão duma licença sabática no ano letivo de 2012/2013) também apoiaram a feitura deste trabalho.

Bibliografia

Branquinho, J. e Murcho, D. 2001. *Enciclopédia de Termos Lógico-Filosóficos*. Publicações Gradiva.

Carnielli, W. e Epstein, R. L. 2008. *Computabilidade, Funções Computáveis, Lógica e os Fundamentos da Matemática*. Editora UNESP, São Paulo.

Copeland, B. J. 2004. *The Essential Turing*. Oxford University Press.

Copeland, B.J. 2017. "The Church-Turing Thesis". *The Stanford Encyclopedia of Philosophy*, Edward. N. Zalta (principal editor), URL = <https://plato.stanford.edu/entries/church-turing/>.

Davis, M. 1958. *Computability and Unsolvability*. McGraw-Hill. Reimpresso com um novo prefácio e apêndice por Dover Publications, Inc. em 1982.

Davis, M. 1965. *The Undecidable*. Raven Press.

Davis, M. 1982. "Why Gödel didn't have Church's thesis". *Information and Control* 54: 3-24.

Davis, M. 2004. *O Computador Universal*. Editorial Bizâncio.

Dawson, J. W. 1997. *Logical Dilemmas*. A K Peters.

Ewald, W. 1996. *From Kant to Hilbert*. Volume II. Oxford University Press.

Ferreira, F. 1992. "Como ser sério com palavras cruzadas". Em *Matemática e Cultura I*, organização de J. Furtado Coelho, 37-53. Centro Nacional de Cultura e Edições Cosmos.

Ferreira, F. 1995. "No paraíso sem convicção... (uma explicação do programa de Hilbert)". Em *Matemática e Cultura II*, organização de J. Furtado Coelho, 87-121. Centro Nacional de Cultura e SPB Editores.

Ferreira, F. 2008. "*Grundlagenstreit* e o intuicionismo brouweriano". *Boletim da Sociedade Portuguesa de Matemática* 58: 1-23.

Gödel, K. e al. 2009. *O Teorema de Gödel e a Hipótese do Contínuo* (2ª edição). Colectânea organizada e traduzida por M. S. Lourenço. Fundação Calouste Gulbenkian.

Goldstein, R. 2009. *Incompletude. A Demonstração e o Paradoxo de Kurt Gödel.* Gradiva.

Hardy, G. H. 1929. "Mathematical proof". *Mind* 149:1-25.

Hilbert, D. 2003. *Fundamentos da Geometria*. Edição revista e coordenada por A. J. Franco de Oliveira. Publicações Gradiva.

Hilbert, D. e Ackermann, W. 1928. *Grundzüge der theoretischen Logik*. Springer.

Hodges, A. 1992. *Alan Turing: the Enigma*. Vintage.

Kahle, R. 2006. "Os teoremas da incompletude de Kurt Gödel". *Boletim da Sociedade Portuguesa de Matemática* 55: 63-76.

Leibniz, G. W. 1960. *Die philosophischen Schriften von Gottfried Wilhelm Leibniz*. Volume VII. Edição de C. I. Gerhardt. Hildesheim: Olms, 1960.

Mancosu, P. 1998. *From Brouwer to Hilbert. The Debate on the Foundations of Mathematics in the 1920s*. Oxford University Press.

Oliveira, A. J. F. 2010. *Lógica e Aritmética* (3.ª edição revista e aumentada). Gradiva.

Pereira, L. M. 2006. "Gödel e a computabilidade". *Boletim da Sociedade Portuguesa de Matemática* 55: 77-90.

Reid, C. 1986. *Hilbert. Courant*. Springer Verlag.

Sernadas, A. e Sernadas, C. 2012. *Fundamentos de Lógica e Teoria da Computação*. College Publications.