Kairos. Journal of Philosophy & Science “16”, “2016”
Center for the Philosophy of Sciences of Lisbon University

A Física da Terminação

José Félix Costa
Departamento de Matemática do
Instituto Superior Técnico
Centro de Filosofia das Ciências (CFCUL)
Universidade de Lisboa, Portugal
fgc@math.tecnico.ulisboa.pt

Sumário. Mostramos que, em virtude dos limites teóricos da computação, nem toda a ciência formulada com carácter preditivo pode ser simulada. Em particular, evidencia-se que a Física Clássica, nomeadamente a Física Newtoniana, padece deste mal, encerrando processos de Zenão.

DOI 10.1515/kjps-2016-0007

1. Introdução

Admitimos que existem leis da natureza, não é? Muitas dessas leis são o fruto da observação cuidada da regularidade e ciclicidade dos fenómenos que ocorrem à nossa volta. Um objecto arremessado de baixo para cima acaba sempre por cair para a terra. À noite sucede-se o dia. Ao dia sucede a noite. O Sol nasce sempre a Este e põe-se sempre a Oeste.

Porque acreditamos que o Sol nascerá amanhã? Somos capazes de jurar que vai nascer. Imaginemos, então, uma moeda imperfeita: em 10 000 lançamentos, não saiu uma só vez “coroa”. Em cerca de 6000×365 dias

o Sol nasceu[1], ao dia sucedeu a noite. Poderemos esperar, com toda a certeza, que nasça amanhã? Ocorre-nos, então uma pequena dúvida... Mas todos nós apostaríamos favoravelmente! De facto, a probabilidade de que o Sol não nasça amanhã é muito pequena. Com certeza ganharemos a aposta. Pois bem, na Física também se faz uma aposta: aplica-se o *princípio da indução*. Os fenómenos da natureza repetem-se sempre que são criadas condições idênticas. Será a Natureza bem comportada? Que natureza têm as leis da Natureza? Estes são problemas da Filosofia da Ciência, entre muitos outros tão ou mais difíceis.

Vejamos em que consiste uma Teoria Física.

O seu primeiro papel é o de síntese: reunir num corpo de definições e relações matemáticas toda a fenomenologia de certo tipo. Por exemplo, há um número infindável de fenómenos elétricos, magnéticos e eletromagnéticos. As equações de Maxwell (conjuntamente com muitas definições e conceitos) são suficientes para os explicar a todos, dentro dos limites da macrofísica. Observamos uma quantidade incomensurável de fenómenos mecânicos: o movimento de corpos à superfície da Terra, o movimento de corpos na atmosfera, o movimento de corpos no sistema solar. Porém, todos eles podem ser explicados, dentro de certos limites macrofísicos, pelas leis de Newton.

Mas há outra preocupação da Física, talvez mais importante: ser capaz de prever... de antecipar a Natureza. O caráter preditivo da Física traduz-se na capacidade de resolver as equações matemáticas a fim de prever o evolvimento dos processos naturais, por exemplo, o estado da atmosfera para amanhã. Para que tal seja possível, torna-se necessário que as equações da Física sejam simuláveis, o que nos leva ao conceito abstrato de computador como um dispositivo que realiza a simulação, ou seja, o dispositivo físico capaz de emular, sob controlo, o fenómeno físico considerado.

O computador vulgar é um dispositivo físico. Será ele capaz de simular (numerica ou simbolicamente) todo o fenómeno físico? Admitindo

1 Há registos de que o Sol nasceu nos últimos 6000 anos.

mesmo que se conhecem *todos* os termos e fatores determinantes da dinâmica do processo? Se não for, então como diz Dante no seu *Inferno*:

Vós que entrais [na Física] perdei toda a esperança.

Haverá alguma relação entre os limites da computação e as leis da Física, aplicadas no limite da realidade sensível? Neste artigo mostramos que certas leis da Física podem não ser simuláveis.

A ser assim... nem toda a Física pode ser usada para prever.

A estrutura deste artigo é a seguinte: introduzimos o conceito de máquina de Turing, um dispositivo abstrato tão ou mais poderoso (!) do que qualquer computador em termos de poder de cálculo; seguidamente discutimos a indecidibilidade da paragem; referimo-nos, depois, a tentativas diversas de resolução do problema da paragem, por exemplo, à introdução do conceito de computador acelerado; finalmente, mostramos que certos processos físicos limite encerram a natureza do problema da paragem.

Usamos no texto alguns símbolos mais raros do que outros. Uns leitores dizem que são matemáticos, outros que, matemáticos ou não, podem ser dispensados. Porém, tais símbolos, num momento ou noutro, poderão resolver uma ambiguidade da leitura.

2. A Máquina de Turing

2.1. A ideia de um computador abstrato

A máquina de Turing foi introduzida por Alan Turing em 1936, num artigo intitulado "On Computable Numbers, with an Application to the Entscheidungsproblem"[2], publicado nos *Proceedings of the London Mathematical Society* (*vide* Turing A., 1936). Algumas correções foram adicionadas pelo próprio Turing e publicadas em 1937 na mesma revista (*vide* Turing A., 1937).

Muitos dos leitores, para quem o conceito de máquina de Turing é familiar, perguntaram certamente a si mesmos: Como chegou Turing ao conceito de máquina de Turing? Turing procurava uma "prova" de que

2 I.e., *Dos números computáveis, com aplicação ao problema da decisão.*

a matemática não é redutível a um procedimento algorítmico. Para fazer uma tal "prova", Turing tinha de encontrar: (a) um conceito razoável de algoritmo e (b) algum problema de decisão em matemática que fosse não algorítmico. E assim teria a demonstração da indecibilidade do *Entscheidungsproblem* de Hilbert, ao tempo um problema matemático em aberto que David Hilbert e Wilhelm Ackermann identificaram em 1928, e David Hilbert (em 1928) adicionou ao seu "Programa de Investigação 1900" para matemáticos[3].

Um algoritmo de decisão não é mais do que um método sistemático para decidir se um dado objeto abstrato, de entre objetos abstratos de certa classe, satisfaz ou não certa propriedade. Por exemplo, decidir se um número natural (expresso na notação decimal) é par. Eis um algoritmo para decidir se um número é par: percorrem-se os dígitos decimais da expressão do número até ao último dígito; se este for *0*, *2*, *4*, *6*, *8*, então pode concluir-se que é par, se não o número não é par, ou seja, é ímpar. Ora um tal problema pode equacionar-se como um problema de decisão: seja *P* o conjunto dos números pares. Saber se um número *n* é par consiste em responder à pergunta:

$$n \stackrel{?}{\in} \mathrm{P}$$

Vejamos como resolveu Turing o problema de encontrar um dispositivo abstrato de computação. Tomou como modelo os chamados computadores humanos dos anos trinta.

As mulheres "computadoras"[4] precisavam de papel (quantidades que podemos pressupor ilimitadas), lápis e borracha. Tomavam notas no papel, apagavam-nas, continuavam a escrever, assentando e apagando até que a computação era dada como terminada. Escrever no papel pode, no limite, considerar-se como escrever sempre na horizontal do papel

3 Embora a origem de tal problema seja o filósofo e cientista Gottfried Leibniz, no final do século XVII, também ele inventor de engenhos de computação.

4 Nos anos 30, em Inglaterra, o termo "computador" significava a pessoa (tipicamente uma mulher) cuja profissão era fazer computações. Uma pessoa podia concorrer a um lugar de computador (*vide* o artigo de W. Barkley Fritz (Fritz, 1996)).

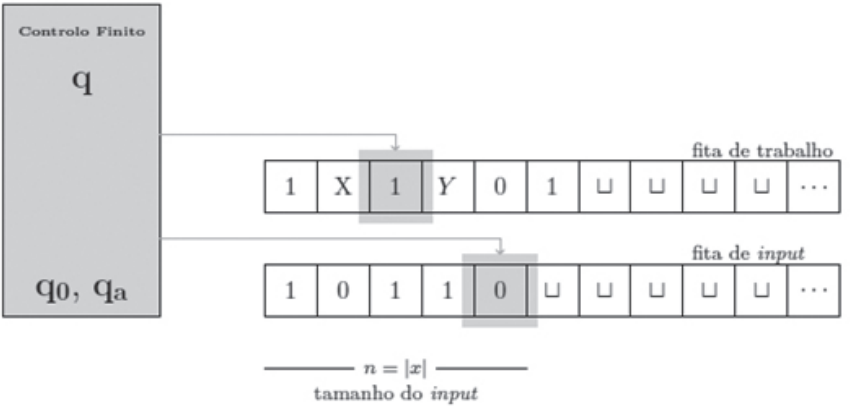

FIGURA 1. Ilustração das fitas e das cabeças de leitura/escrita de uma máquina de Turing. Algumas particularidades: (a) a máquina representada tem duas fitas, (b) o alfabeto de input, denotado por Σ, é binário, e (c) o alfabeto de trabalho, denotado por Γ, é maior, $0,1,X,Y,\sqcup$, nomeadamente contém o símbolo branco $\sqcup$. A fita de *input* é exclusivamente de leitura.

quadriculado, ao longo de uma fita ilimitada de quadrículas – as *células* ou *casas*. É um problema de adaptação ao uso de papel de serpentina para cálculos... Certamente que nos adaptaríamos... Em cada momento da computação, o computador tem acesso a informação finita; baseado no seu estado mental[5], o computador observa a informação a que tem acesso e realiza uma transição para um novo estado mental, possivelmente apagando a informação lida e substituindo-a por informação nova, e movendo-se para a nova página ou a anterior. Enfim... andando para a frente ou para trás ao longo da serpentina. Uma descrição mais completa de como a ideia germinou pode ser encontrada no livro de Martin Davis (*vide* Davis, 2000). Este processo de computação humana foi abstraído por Turing da maneira que se descreve a seguir.

Na Figura 1, vemos a representação pictórica de uma máquina de Turing com duas fitas, a fita de input e uma fita de trabalho, conjuntamente

5 Era estado mental que se designava e ainda designa. Esta terminologia advém do facto de o computador, de alguma forma, ter sido desenhado à imagem de processos cognitivos.

com o controlo finito. O controlo finito é apresentado num diagrama de transições que discutiremos mais à frente. Na fita de input, vemos uma palavra binária escrita nas primeiras *n* células da fita: dizemos que o *input* tem tamanho *n* (= *5* neste caso), não importando o alfabeto usado para o escrever. Na fita de trabalho vemos uma palavra escrita com letras de um alfabeto maior que ocupa *6* células. Este alfabeto maior é designado por alfabeto de trabalho: a máquina trabalha com o alfabeto de *input*, possivelmente enriquecido com mais símbolos, considerados necessários na especificação/execução do algoritmo. As cabeças de leitura/escrita podem ler/escrever um símbolo de cada vez. De facto, poderíamos ter definido a máquina de Turing de tal maneira que a cabeça pudesse ler/escrever vários símbolos ao mesmo tempo. Mas se tal função poderia melhorar a linguagem de especificação de algoritmos, não muda a eficiência da máquina. O controlo finito é um dispositivo finito de estado/transição – chamado autómato finito – que descreve o algoritmo.

A memória da máquina está dividida em duas partes: uma é a memória externa, a informação que a máquina tem nas fitas, e a outra é a memória que a máquina tem no controlo finito que não pode mudar durante as computações[6]. Há três estados notáveis representados: o estado q em que a máquina se encontra, o estado inicial q_0 e um estado de paragem/aceitação q_a. Assim, esperamos encontrar no controlo finito – no diagrama de transições – o que é requerido à máquina fazer, e.g., quando se encontra no estado q, a cabeça de leitura da fita de input está a ler *0* e a cabeça de leitura/escrita da fita de trabalho está a ler *1* (Figura 1). A máquina de Turing está completamente especificada se, para todos os estados e para todos os símbolos sob as cabeças de leitura/escrita, estiver especificado o que fazer a seguir. Quando a máquina atinge um estado de paragem, seja ele o de aceitação, seja ele o de rejeição, q_a ou q_r, respetivamente, desliga-se. Não é, pois, muito importante especificar o que a máquina tem a fazer num estado de paragem.

6 E.g., as palavras podem ser memorizadas de uma só vez no controlo finito, de modo a que a máquina de Turing pode escrevê-las na fita sempre que necessário – esta é a memória fixa.

2.2. Configurações

Uma fita é uma sequência unidirecional e infinita de *células* ou *casas* onde se inscrevem símbolos de um alfabeto Γ (finito) que contém um símbolo especial branco ▱ . Desta forma, a fita abstrai o papel branco em quantidades ilimitadas usado por um computador humano para realizar computações. As notas que o computador decide escrever encontram-se alinhadas nas casas da fita e são seguidas por um número infinito de casas brancas. Podemos mesmo agrupar as células em *k* fitas, às quais juntamos uma fita de *input* que contém os dados iniciais e outra de *output* para imprimir o resultado da computação se tal for necessário.

A configuração da máquina corresponde a *k+2* sequências de símbolos de Γ divididas pelo estado de *Q* em duas subsequências, a subsequência de símbolos à esquerda e a subsequência de símbolos à direita da cabeça de leitura/escrita; a segunda subsequência inclui o símbolo sob a cabeça de leitura/escrita.

Assim, a configuração da máquina de Turing representada na Figura 1, é o par de sequências *1011q0* e *1Xq1Y01*. Estas duas sequências contêm toda a informação de que necessitamos saber sobre a máquina, neste passo da computação: o estado mental da máquina, o conteúdo das fitas e a posição das cabeças de leitura/escrita.

2.3. Definição formal de máquina de Turing

Em termos mais rigorosos pode definir-se uma máquina de Turing *M* através dos seus elementos:

$$\mathbb{M} = (k, \Sigma, \Gamma, \mathrm{Q}, \mathrm{q}_0, \mathrm{q}_a, \mathrm{q}_r, \delta)$$

Onde:

1. *Q* é um conjunto finito de estados;
2. $k>0$ é o número de fitas de trabalho de M (em adição a uma fita exclusivamente de leitura e a uma fita exclusivamente de escrita);
3. Σ é o conjunto finito dos símbolos que podem ser lidos na fita de *input* e escritos (ou impressos) na fita de *output*; não contém o símbolo especial "branco" ▱;

4. Γ é o conjunto dos símbolos que a máquina pode ler e escrever nas fitas de trabalho, tal que $\square \in \Gamma$, e que contém todos os símbolos de Σ;
5. δ é a função de transição: sabendo qual o estado mental $p \in Q$, sabendo quais os $k+1$ símbolos de Γ lidos, $u_0, \ldots, u_k$, inscritos nas casas sob as cabeças de leitura/escrita das $k+1$ fitas (fita de input e k fitas de trabalho), a máquina transita para o estado mental q, possivelmente reescreve os símbolos, $u'_1, \ldots, u'_k$ (excetua-se o símbolo de input, $u'_0 = u_0$), possivelmente escreve na respetiva casa da fita de *output* u'_{k+1}, e faz mover cada uma das $k+2$ cabeças de leitura/escrita (a da fita de *input*, as das k fitas de trabalho e a da fita de *output*), uma casa para a direita ("R"), uma casa para a esquerda ("L"), ou deixa-a na mesma casa ("N").
6. $q_0 \in Q$ é o estado inicial;
7. $q_a \in Q$ é o estado de aceitação;
8. $q_r \in Q$ é o estado de rejeição (q_0, q_a e q_r são distintos).

Às vezes tornamos as máquinas de Turing mais simples especificando-as apenas com uma fita, onde o *input* se encontra inscrito nas células mais à esquerda, tal que a cabeça pode ler e escrever e, consequentemente, reescrever o *input*. Porém, as máquinas de Turing multifita com fitas de input e *output* constituem um modelo muito útil quando se pretende construir uma máquina de Turing em módulos.

Para representar a função transição δ recorremos a um diagrama. Neste diagrama, os círculos representam os estados e as setas representam as transições. Vamos analisar as transições do diagrama da Figura 2: por exemplo, temos uma transição entre os estados q_0 e q_1, etiquetada por $A \rightarrow X,R$:

> Encontrando-se a máquina no estado inicial q_0, se a cabeça de leitura/escrita está a ler A, então a máquina executa uma transição para o estado q_1, escrevendo X onde está A e deslocando a cabeça de leitura/escrita uma casa para a direita ("R" do inglês *right*).

Consideremos agora a transição entre os estados q_2 e q_3, etiquetada por $\square \rightarrow \overline{B}$,*L*.

> Encontrando-se a máquina no estado inicial q_2, se a cabeça de leitura/escrita está a ler branco $\square$, então a máquina executa uma transição para o estado q_3, escrevendo $\overline{B}$ na casa branca e deslocando a cabeça de leitura/escrita uma casa para a esquerda ("L" do inglês *left*).

2.4. Computações

Um passo de uma computação da máquina de Turing M na configuração c_i produz a nova configuração c_f. Começando na configuração inicial c_0, a máquina *M* gera uma sequência possivelmente infinita de configurações. A máquina de Turing, provida de *input*, ou (a) nunca para, trabalhando para sempre, saltando de configuração em configuração, não necessariamente num ciclo em que repita configurações, ou (b) para num número finito de passos numa configuração que contém o estado de aceitação q_a ou o estado de rejeição q_r.

A computação de uma máquina de Turing é a sequência finita ou infinita das suas configurações.

Consideraremos máquinas de Turing que podem não parar para alguns, ou mesmo todos, os *inputs*.

Relembremo-nos de que Σ é o alfabeto com se escrevem as palavras que servem de *input* à máquina, tal como na Secção 2.3. O símbolo Σ* denota o conjunto de todas as palavras que se escrevem com as letras do alfabeto Σ – isto é, o conjunto de todos os possíveis *inputs*. Uma *linguagem*, digamos *A*, é um conjunto finito ou infinito de palavras de Σ*:

Definição 1. Um conjunto *A* é *decidido* por uma máquina de Turing *M* se a computação de *M* termina no estado de aceitação sempre que $w \in A$ e termina no estado de rejeição sempre que $w \notin A$.

A Definição 1 introduz o chamado *problema de decisão*, isto é *o problema de encontrar um algoritmo*, isto é, uma máquina de Turing *M*, que decide a questão

$$w \in A \; ?$$

Este predicado lê-se "*w* pertence a *A*?" Em caso afirmativo, a máquina de Turing *M*, mais tarde ou mais cedo, deverá "responder" que sim, transitando para o estado de aceitação. Em caso negativo, a máquina de Turing deverá "responder" que não, transitando para o estado de rejeição. Em qualquer dos casos, a máquina deverá parar.

Um outro problema de quase decisão, ou de semidecisão, consta do seguinte: suponhamos que *A* é o conjunto dos *inputs* para os quais a máquina *M* para no estado de aceitação; que possíveis inputs não estão em *A*? Temos os inputs para os quais a máquina *M* para no estado de rejeição e os *inputs* para os quais a máquina *M* não para. Diz-se então que *A* é conjunto reconhecido por *M*. Se *M* para para todos os *inputs*, então o conjunto *A*, que é o conjunto reconhecido por *M*, também é o conjunto decidido por *M*, de acordo com a Definição 1 (nestas circunstâncias, *reconhecível = decidível*).

Definição 2. Um conjunto *A* é reconhecível por uma máquina de Turing *M* se a computação de *M* relativamente ao *input w* termina no estado de aceitação se e só se $w \in A$.

Para concluir esta exaustiva introdução de conceitos, definimos *função computável* e *número real computável*.

Definição 3. Uma função (total) dos números naturais $n \in \mathbb{N}$ para os números naturais $f(n) \in \mathbb{N}$ diz-se *computável* se existir uma máquina de Turing *M* que, para todo o *input* $n \in \mathbb{N}$[7] escreve *f(n)* na fita de *output*[8] antes de aceitar e desligar-se.

Neste caso, a máquina de Turing, para além de transitar para o estado de aceitação, deverá ter escrito na fita de *output* o valor da função computada. E.g., se a máquina está a calcular o valor de 3×5,[9] ela deverá, antes de desligar-se, escrever 15[10] na fita de *output*[11].

7 Por exemplo, expresso como palavra de algarismos decimais.

8 Idem.

9 Por exemplo, usando a notação que será introduzida na secção seguinte, *111x11111*

10 *111111111111111*.

11 Note que a máquina de Turing em causa é a mesma para todos os inputs, o que pode variar é o input para a máquina.

Definição 4. Um número real $r \in \mathbb{R}$ diz-se *computável* se existir uma máquina de Turing *M* que, para todo o *input* $n \in \mathbb{R}$, escreve os primeiros n dígitos (e.g., decimais) de *r* na fita de *output* antes de aceitar e desligar-se.

Existe apenas um número contável infinito de funções computáveis (tantas quantas os números naturais); porém, o número das funções não computáveis é não contável (digamos, tantas quantas os números reais). (I.e., a maior parte das funções são não algorítmicas.) Mais, a máquina de Turing não pode computar funções com crescimento arbitrariamente grande. Existe um limite de crescimento para as funções computáveis.

2.5. Exemplos de máquinas de Turing

Seguem-se agora dois exemplos de máquinas de Turing, uma especificada para sumariamente ilustrar o diagrama de transições e outra para resolver um problema de decisão relacionado com a divisão.

Na especificação de uma máquina de Turing complexa é por vezes necessário fazer uma cópia de uma sequência de símbolos. Na Figura 2, mostramos o diagrama de transições de uma máquina que realiza esse propósito.

A máquina de Turing *M* inicia a sua computação na primeira célula (a célula mais à esquerda) da sua única fita, onde, nas primeiras /*w*/ células (/*w*/ representa o número de letras da palavra *w*) podemos encontrar o *input w* escrito com *A*'s e *B*'s; *M* completa a cópia na célula mais à esquerda da sequência *ww*, i.e., *M* para com duas cópias de *w* sem espaço entre elas e com a cabeça de leitura/escrita no seu lugar original, o primeiro símbolo de *ww*.

A máquina de Turing procede como se segue: lê *A*, substitui *A* por *X* escreve $\overline{A}$ na primeira célula branca à direita; a cabeça de leitura/escrita volta atrás até à primeira célula que encontrar marcada com *X* e avança uma casa para a direita; a máquina lê *B*, substitui *B* por *Y* e escreve $\overline{B}$ na primeira célula branca à direita; a cabeça de leitura/escrita volta atrás até à primeira célula que encontrar marcada com *Y* e avança uma casa para a direita; assim que os *A*'s e os *B*'s estiverem esgotados, a máquina reescreve cada *X* e cada $\overline{A}$ num *A* e cada *Y* e cada $\overline{B}$ num *B*.

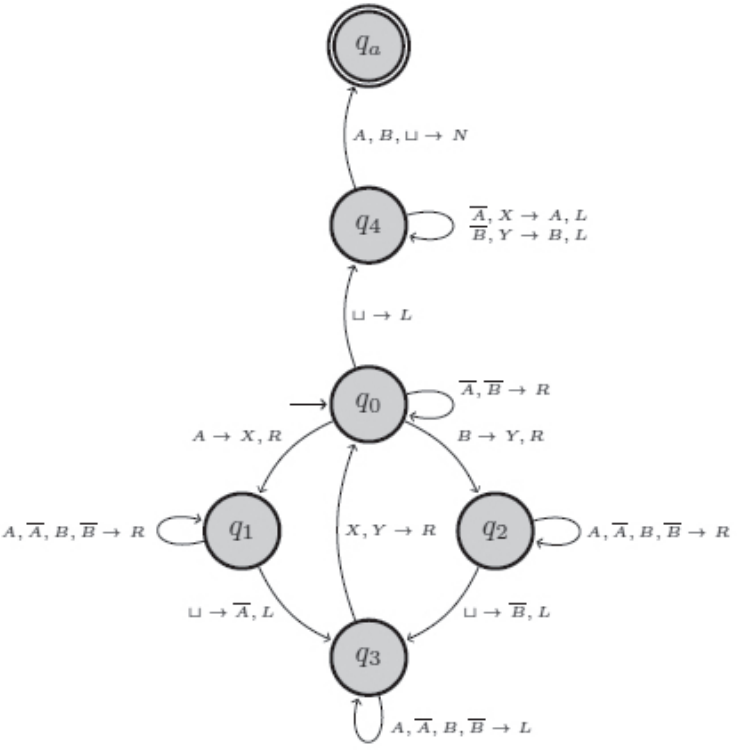

FIGURA 2. Exemplo de máquina de Turing – copiador. O estado inicial q_0 é identificado através de uma seta sem origem mas com destino e o estado de aceitação é identificado com um duplo círculo. O estado de rejeição não está representado por não ser relevante. Todas as transições possíveis que não estão indicadas são para o estado de rejeição.

Este exemplo dá a ideia de computação de uma máquina de Turing. O leitor pode exercitar-se especificando máquinas de Turing que computem as operações triviais da aritmética, com os seus argumentos em unário[12] e quaisquer dois argumentos separados pelo símbolo da operação, tal como em *#111+11* or *#111×11*. Pode supor-se que, no início, a fita da máquina contém, digamos, *#111+11* (o que denota *3+2*); a máquina deverá operar de modo a que, no fim da computação, a palavra *#11111* seja encontrada nas células mais à esquerda da fita; mais (!),

12 I.e., o número *n é* denotado pela sequência de *n 1's*. Outro modo de fazer isto, para evitar a sequência vazia que denota o número 0, é representar n pela sequência de *n+1 1's*.

o programa deverá funcionar não importando o tamanho dos números a adicionar (*#1…1+1…1*).

Para estes fins, deverá o leitor examinar detalhadamente o diagrama de transições da Figura 2. Deverá percorrê-lo desde o estado inicial ao estado de aceitação, pressupondo determinada palavra *w* de *A*'s e *B*'s a copiar.

Um segundo exemplo de máquina de Turing resolve o problema da divisão da seguinte maneira: *n* repetições de uma letra denota o número *n* e a palavra $\#a^m b^n c^{n \div m}$ especifica que *n b*'s a dividir por *m a*'s dá $n \div m$ *c*'s; o símbolo # marca o início da palavra. Note que a divisão por zero não está permitida, pelo que nenhuma das palavras $\# \; b^n c^{n \div 0}$ pode ser aceite; porém, todas as palavras $\#a^m$ ($m \neq 0$) são aceites, denotando que $0 \div m = 0$, para todo o $m \neq 0$. No início da computação, o *input* é qualquer palavra que se escreva com os símbolos *a*, *b*, *c* e #.

Eis, na Figura 3, uma máquina de Turing que resolve o problema. A máquina marca com *X* cada uma das letras do divisor, com *Y* cada uma das letras do dividendo e com *Z* cada uma das letras do quociente.

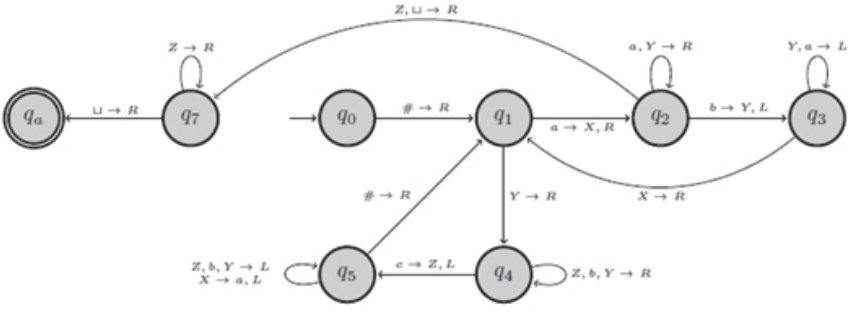

FIGURA 3. Máquina de Turing: divisor. De novo, o estado inicial q_0 é identificado através de uma seta sem origem mas com destino e o estado de aceitação é identificado com um duplo círculo. O estado de rejeição não está representado por não ser relevante. Todas as transições possíveis que não estão indicadas são para o estado de rejeição.

O resultado da divisão é o número de vezes que a palavra que serve de divisor "cabe" na palavra que serve de dividendo.

3. Indecidibilidades

As máquinas de Turing são objetos matemáticos finitos: dois alfabetos finitos, um conjunto finito de estados, três estados conspícuos e uma função δ que aplica um conjunto finito noutro conjunto finito. Todos estes objetos podem, de uma forma trabalhosa mas que não deixa de ser trivial, ser codificados através de sequências de *0*'s e *1*'s. Essencialmente, o resultado deste esforço de codificação, se for bem feito, é o seguinte: toda a palavra binária representa uma máquina de Turing e toda a máquina de Turing pode ser codificada em binário.

Podemos, assim, olhar para uma palavra binária como um dado ou como o código de uma máquina de Turing. Se *w* é uma palavra binária, então por *<M,w>* denotamos o código da máquina M que tem registado na sua fita de *input* a palavra *w*. Note-se que *<M,w>*, como código que é, também é uma palavra binária. E como palavra binária... pode ser dada como *input* a outra máquina de Turing *M'*. Nada há de especial em considerar *<M,w>* como *input* de *M'*, tal como nada há de especial em dar a um computador, como *input*, um programa escrito numa determinada linguagem de programação, conjuntamente com os dados para esse programa.

Assim, podemos ter tarefas dos seguintes tipos: (a) dada uma máquina de Turing *M*, determinar o seu código binário *<M>*, (b) dada uma máquina de Turing *M*, determinar o código binário de *M* com o *input w*, o que se denota por *<M,w>*, (c) dada a máquina de turing *M*, escrever o código binário de *M* munida do *input* *<M>* (!), o que se escreve *<M,<M>>*.

Teorema 1. (Máquina de Turing universal) Existe uma máquina de Turing universal *U* que recebe como *input* *<M,w>*, o código binário de uma máquina de Turing *M* e uma palavra binária *w*, tal que, para toda a máquina *M* e para toda a palavra *w*, simula *M* quando o *input* é *w*:

$$U \text{ com input } \langle M,w\rangle \equiv M \text{ com input } w$$

O que se segue nesta secção são provas de indecidibilidade de problemas de decisão.

Definição 5. O conjunto de aceitação *A* é a coleção de todos os códigos *<M,w>*, tais que *M* é uma máquina de Turing que aceita o *input w*.

Definição 6. O problema da aceitação "*<M,w>∈A*" é o problema de decidir dado o código *<M,w>*, se a máquina de Turing *M* aceita ou não o *input* w.

Para mostrar que este problema é indecidível por uma máquina de Turing, digamos por um computador convencional, vamos usar um raciocínio lógico de redução ao absurdo. Uma prova semelhante, mas mais extensamente comentada, pode encontrar-se no célebre livro de divulgação de Roger Penrose (*vide* Penrose, 1989).

Vejamos, então, a prova de um já famoso problema matemático da computação:

Teorema 2. O problema da aceitação "*<M,w>∈A*" é indecidível por máquina de Turing.

A demonstração deverá ser seguida com extrema atenção, pois quem a não conhece, apesar de ser uma prova pequena, tem, em geral, dificuldade em entendê-la.

3.1. O problema da aceitação

Suponhamos que o problema "*<M,w>∈A*" pode ser decidido por máquina de Turing $\mathbb{A}$.

Ter-se-ia: $\mathbb{A}$ aceita a palavra *<M,w>* se *M* aceita o *input w* e $\mathbb{A}$ rejeita a palavra *<M,w>* se *M* não aceita o *input w*. Esta asserção advém da definição de decisor $\mathbb{A}$ e da definição do conjunto da aceitação da Definição 5.

A partir da máquina $\mathbb{A}$, construímos uma outra máquina $\tilde{\mathbb{A}}$ que funciona da seguinte maneira: $\tilde{\mathbb{A}}$ aceita *<M>* no caso de *M* não aceitar *<M>* e $\tilde{\mathbb{A}}$ rejeita *<M>* se *M* aceita *<M>*. *É dito: construímos. Resta saber como...* No entanto, vejamos se entendemos o que se pretende. O que é não aceitar? Não aceitar não significa necessariamente rejeitar. Significa rejeitar ou não parar. A construção da máquina $\tilde{\mathbb{A}}$ a partir da máquina $\mathbb{A}$ não oferece grande dificuldade: dado o *input <M>*, a máquina $\tilde{\mathbb{A}}$ usa o código de $\mathbb{A}$ para saber se *M* aceita ou não aceita *<M>* e rejeita ou aceita de acordo com o resultado, respetivamente.

Agora segue-se o desenlace da demonstração:

Perguntemo-nos: Será que $\tilde{\mathbb{A}}$ aceita <$\tilde{\mathbb{A}}$>? Mais uma vez, recordemos que $\tilde{\mathbb{A}}$ pretende ser um decisor[13] e que <$\tilde{\mathbb{A}}$> é já uma palavra, o código do procurado decisor $\tilde{\mathbb{A}}$.

Se $\tilde{\mathbb{A}}$ aceita <$\tilde{\mathbb{A}}$>, então rejeita <$\tilde{\mathbb{A}}$>, o que é uma contradição. *Relembremos outra vez a função de* $\tilde{\mathbb{A}}$...

Terá, então, de verificar-se que $\tilde{\mathbb{A}}$ rejeita <$\tilde{\mathbb{A}}$>, ou seja, que $\tilde{\mathbb{A}}$ aceita <$\tilde{\mathbb{A}}$> o que também é absurdo. *Relembremos outra vez a função de* $\tilde{\mathbb{A}}$...

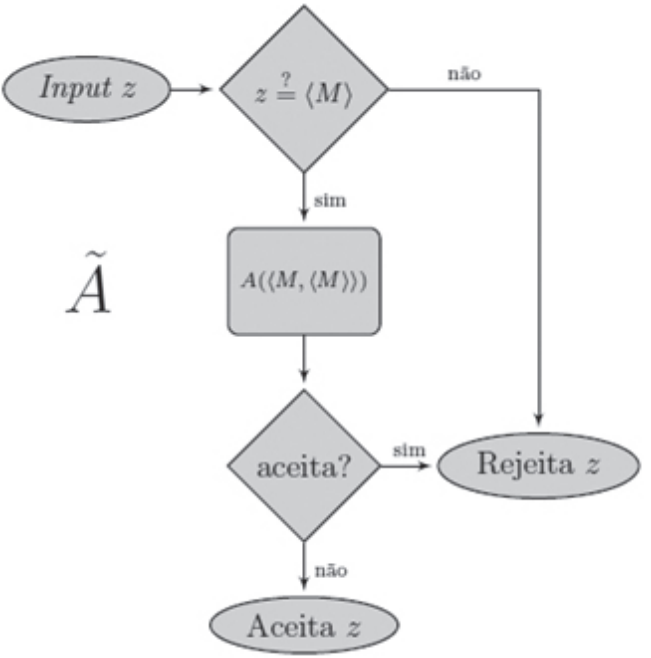

FIGURA 4. Este é o programa $\tilde{\mathbb{A}}$. A impossibilidade de existência de $\mathbb{A}$ mostra-se quando se dá a $\tilde{\mathbb{A}}$ o *input z* = <$\tilde{\mathbb{A}}$ >, isto é quando se toma *M*=$\tilde{\mathbb{A}}$. Os losangos denotam decisões: primeiramente, a máquina verifica se o *input z* codifica uma máquina de Turing *M*; mais à frente testa se $\mathbb{A}$, procedimento que supostamente para para todos os *inputs*, aceita o *input <M,<M>>*. O retângulo denota o procedimento $\mathbb{A}$ a operar sobre o seu *input*.

13 Para um decisor *não aceitar = rejeitar*. Para uma máquina de Turing arbitrária não *aceitar = rejeitar ou não parar*.

Quer isto dizer que $\tilde{\mathbb{A}}$ não aceita nem rejeita $<\tilde{\mathbb{A}}>$, o que é absurdo, uma vez que $\tilde{\mathbb{A}}$ é um decisor e, consequentemente, tem de decidir sobre todos os *inputs*.

A Figura 4 ilustra o procedimento $\tilde{\mathbb{A}}$ descrito nos parágrafos anteriores. Este procedimento é uma especificação informal da máquina de Turing $\tilde{\mathbb{A}}$ que, na suposição de que a máquina de Turing $\mathbb{A}$ existe, decide o conjunto $\{<M>: M$ é uma máquina de Turing que não aceita $<M>\}$. Suponha que se dá como *input* ao procedimento da Figura 4 a palavra $<\tilde{\mathbb{A}}>$. Seguindo o fluxo dos dados no diagrama, concluímos que $\tilde{\mathbb{A}}$ aceita $<\tilde{\mathbb{A}}>$ se e só se $\tilde{\mathbb{A}}$ não aceita $<\tilde{\mathbb{A}}>$, o que é uma contradição.

Ora a máquina $\tilde{\mathbb{A}}$ é construída à custa de $\mathbb{A}$. Se a máquina é impossível é porque a própria máquina $\mathbb{A}$ não pode existir. Conclui-se que o problema da aceitação não pode ser decidido por uma máquina de Turing.

3.2. O problema da paragem

O problema da paragem pode agora ser demonstrado indecidível, também por absurdo.

Definição 7. O conjunto de paragem H é a coleção de todos os códigos $<M,w>$, tais que M é uma máquina de Turing que para (em aceitação ou rejeição) quando o *input* é w.

Teorema 3. O problema da paragem "$<M,w>\in H$" é indecidível por máquina de Turing.

Antes de mais, convém fazer notar que tal problema de decisão – a paragem – pode ser justamente equacionado, uma vez que há máquinas de Turing que manifestamente não param: por exemplo, a máquina de transição única abreviada $\cdots\rightarrow\cdots$, R, isto é, não importando o símbolo lido, a cabeça de leitura move-se uma casa para a direita; outro exemplo, a máquina de duas transições $\cdots\rightarrow\cdots$, R e $\cdots\rightarrow\cdots$, L, isto é, a máquina que, não importando qual o símbolo lido move a cabeça uma casa para a direita e, depois, uma casa para a esquerda. Porém, certas máquinas podem parar para certos *inputs* e não parar para outros *inputs*, ou seja, pode acontecer que, para $w_1 \neq w_2$, se tenha $<M,w_1>\in H$ e $<M,w_2>\notin H$.

Ou seja, decidir se uma máquina de Turing para ou não para para certo *input* é um problema que pode muito bem ser equacionado para máquinas de Turing. Na vida prática do programador, tal problema corresponde a, dado um programa arbitrário, escrito numa certa linguagem, saber de antemão, sem o executar, se o programa origina, através de um ciclo, para certo *input*, computações infinitas.

O que pode ser perguntado é se o problema da paragem é afinal, um problema de má programação... Não é, como vamos poder comprovar na próxima Secção 5.

Procede-se assim. Suponha que que o problema da paragem tem decisor $\mathbb{H}$ (uma máquina de Turing). Se demonstrarmos que através de $\mathbb{H}$ podemos decidir a aceitação, estamos a demonstrar também que o decisor $\mathbb{H}$ não pode existir, pois a aceitação, como vimos atrás, na Secção 3.6, não é decidível.

De facto, a existência do decisor $\mathbb{H}$ implica a existência do decisor $\mathbb{A}$. Vejamos porquê.

Será que *M* aceita *w*? Tomemos o decisor com *input* $\langle M,w\rangle$. Se $\mathbb{H}$ aceita, então concluímos que *M* para para o *input w*. Pelo que podemos executar a máquina *M* com *input w*, com toda a certeza de que a computação chegará ao fim. Se a máquina *M* aceitar *w*, então concluímos que $\langle M,w\rangle \in A$, caso contrário concluímos que $\langle M,w\rangle \notin A$. Se $\mathbb{H}$ rejeita $\langle M,w\rangle$, então concluímos que *M* não para para o *input w* e, consequentemente, *M* não aceita *w*, ou seja, $\langle M,w\rangle \notin A$.

A Figura 5 ilustra o procedimento para decidir o problema da aceitação, descrito no parágrafo anterior. Este procedimento é uma especificação informal da máquina de Turing $\mathbb{A}$ que, na suposição de que a máquina de Turing $\mathbb{H}$ existe, decide o conjunto *A*.

Conclui-se que, a existir tal decisor $\mathbb{H}$, existe também o decisor $\mathbb{A}$ para o problema da aceitação. Mas o problema da aceitação, como vimos, não tem decisor (em termos de máquinas de Turing). Consequentemente, o problema da paragem não pode ter decisor.

Devemos agora ler de novo esta Secção 3.7, partindo do pressuposto de que o decisor $\mathbb{A}$ não pode existir. Depois deveremos revisitar a Secção 3.6 para verificar que entendemos que tal decisor $\mathbb{A}$ não pode mesmo existir.

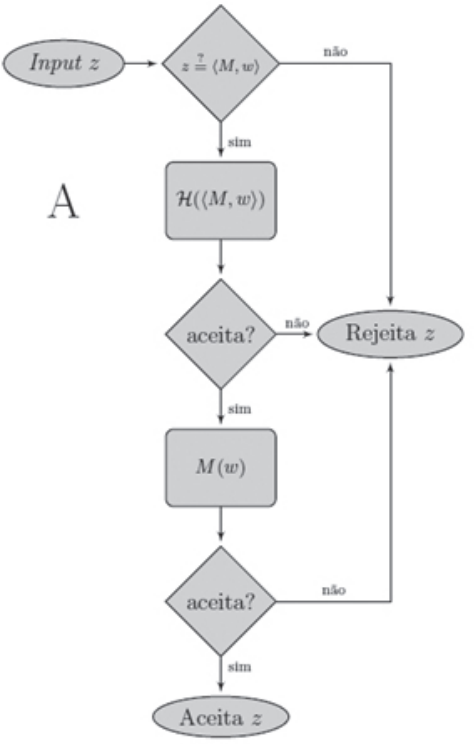

FIGURA 5. A impossibilidade de existência de $\mathbb{H}$ mostra-se reduzindo $\mathbb{A}$ a $\mathbb{H}$: se $\mathbb{H}$ fosse decidível, $\mathbb{A}$ também o era. Os losangos denotam desisões: primeiramente, a máquina verifica se o *input z* codifica uma máquina de Turing *M* e um *input w* (para *M*); mais à frente, testa se $\mathbb{H}$, procedimento que para para todos os *inputs*, aceita o *input* <*M,w*>; por fim, verifica se *M* aceita *w*. Os retângulos denotam os procedimentos $\mathbb{H}$ e *M* operando sobre os seus respectivos *inputs*.

Assim, nesta secção, vimos que problemas como a *aceitação* ou a *paragem* não podem resolver-se com auxílio de máquinas de Turing e, consequentemente, através dos computadores convencionais.

E quanto aos outros computadores... os não convencionais? Poderão eles resolver a paragem? As próximas secções procurarão ser muito elucidativas a este respeito. Há investigadores, em todo o mundo, a procurar demonstrar que, de um modo ou de outro, algum sistema físico, ou químico, ou outro, pode ser construído para resolver o problema da paragem!

4. Conjetura de Collatz e Predicados Π2

Suponhamos que nos é dada uma função *f* que opera sobre números naturais para dar números naturais. E.g., a função *f(n) = 3n +1*. Dado o número *2*, a função retorna o número 7. A função pode agora ser aplicada a *7*, retornando o número *22*. Depois a função pode ser aplicada a *22* retornando *67*. Escrevemos *f f f(2) = 67*. Em geral

$$f\ f\ f\ \cdots f(n)$$

diz-se a função iterada de *f*: dados o número natural *n* e o número de vezes *m* que a função *f* deve ser iterada, obtém-se o resultado

$$\underbrace{f\ f\ f\ \cdots f(n)}_{m\ vezes}$$

Suponhamos agora que, dado o *input n*, se verifica, depois de iterar uma certa função *f*, não necessariamente a anterior, *m* vezes, que

$$\underbrace{f\ f\ f\ \cdots f(n)}_{m\ vezes} = 1$$ [14]

Podemos escrever:

$$existe\ m\ tal\ que \quad \underbrace{f\ f\ f\ \cdots f(n)}_{m\ vezes} = 1$$

Para verificar que assim é, apenas temos de iterar *f* até obter *1*. Tarefa que pode prolongar-se indefinidamente...

Fazemos a seguinte conjetura a respeito de *f*:

$$qualquer\ que\ seja\ n, existe\ m\ tal\ que \quad \underbrace{f\ f\ f\ \cdots f(n)}_{m\ vezes} = 1$$

14 O valor 1 é aqui dado como exemplo

Tal asserção diz-se um predicado Π_2 (lê-se *pi 2*). Assusta pensar num objeto de nome predicado pi 2, mas não é assim tão complicado: estamos a afirmar que *para todo o número natural n*, *input* de *f*, iterando *f* suficientes vezes, *m* vezes, obtém-se *1*:

$$\underbrace{f\ f\ f\ \cdots f(n)}_{m\ vezes} = 1$$

Ora... onde está o problema da paragem envolvido nisto?

Suponhamos que a máquina de Turing *M* verifica, para todo o *n*, onde *n* é o input, se existe um *m* tal que

$$\underbrace{f\ f\ f\ \cdots f(n)}_{m\ vezes} = 1$$

Este predicado Π_2 é verdadeiro se e só se a máquina *M* parar para todos os *inputs*. O problema não pode, em geral, resolver-se computacionalmente, por duas razões: (a) o problema de se saber, com generalidade, se uma máquina de Turing para para todos os inputs é indecidível e (b) o problema de se saber, com generalidade, se uma máquina de Turing para para um só *input* também é indecidível.

Vamos considerar agora a função *f* definida assim:

$$f(n) = \begin{cases} n/2, & n\ \text{é}\ par \\ 3n+1, & n\ \text{é}\ ímpar \end{cases}$$

Trata-se de uma função deveras simples. Designemo-la por *função de Collatz*. A função *f* pode ser iterada. Como a máquina de Turing *C* que resolve este problema de iteração da função de Collatz é um pouco complicada, escrevemos antes um programa de computador numa linguagem familiar aos programadores (o que é, de facto, equivalente):

Iterando a função de Collatz:

Input n;
While n ≠ 1 **Do If** even(n) **Then** n := n/2 **Else** n := 3n + 1

Sequências de números produzidos durante a execução do programa para os *inputs 4*, *5* e *7*, respetivamente:

4, 2, 1 **ACEITA**

5, 16, 8, 4, 2, 1 **ACEITA**

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 **ACEITA**

Ninguém sabe se a iterada da função de Collatz dá *1* em todos os casos. Se o *problema da paragem para todo o input* tivesse solução computacional *M*, submeteríamos a máquina de Turing *C* à máquina de Turing *M*, em consequência do que teríamos resposta para o nosso problema: se a resposta fosse afirmativa, saberíamos que a máquina para para todos os *inputs*; se a resposta fosse negativa, então saberíamos que, para certo valor do input *n*, a máquina de código *<C,n>* não para; bastaria, então, submeter ao decisor *H* da Secção 3 as sucessivas máquinas <C,1>, *<C,2>*, *<C,3>*, *<C,4>*, ..., *<C,m>*, ..., até encontrar o primeiro número *m*, para o qual a iterada da função de Collatz não dá 1.

Queremos com isto concluir que o problema da paragem não é somente um problema à volta de *debugging* de *software*, isto é, um problema à volta da dificuldade em eliminar ciclos infinitos em programas de computador devidos a descuidos do programador. É antes um problema matemático que resulta da incapacidade computacional de provar que a iterada de Collatz dá sempre *1* e, sobretudo, que as iteradas das funções (computáveis), em geral, dão certo valor prefixado.

A função que considerámos está relacionada com uma conjetura matemática ainda não demonstrada: a Conjetura de Collatz (de Lothar Collatz, embora tenha muitos outros nomes, entre os quais Conjetura *3n+1*, Conjetura de Ulam (de Stanislaw Ulam), Problema de Kakutani (de Shizuo Kakutani), Conjetura de Thwaites (de Sir Bryan Thwaites), Algoritmo de Hasse (de Helmut Hasse), Problema de Siracusa). Diz o seguinte a nossa conjetura: *não importa qual é o input, iterando suficientemente a função de Collatz, chega-se sempre a 1*.

Pois bem: nenhum matemático conseguiu até hoje demonstrar ou refutar a Conjetura de Collatz. Para o fazer, podemos aplicar as técnicas que bem entendermos.

Mais, o problema de Collatz pode ajudar-nos a examinar, mais aprofundadamente, a natureza do problema da paragem. O problema da paragem, como vemos, está associado à capacidade das máquinas de Turing de realizarem computações infinitas, ou seja ao tempo da computação, entendido como número de transições que a máquina executa até se desligar.

Olhemos agora para o espaço que a máquina consome, isto é, o número de casas que são visitadas no decurso de uma computação.

5. Mais Sobre o Problema da Paragem

Na iteração da função de Collatz, como vimos, o padrão de comportamento da iterada, e, portanto, da máquina de Turing *C*, é o seguinte: os números desatam a crescer até certa magnitude, oscilam e, depois, decrescem até *1*. Os números crescem desmesuradamente, não se lhes pode impor um limite de crescimento. Quanto maiores são os números, maior é o espaço necessário para os escrever, em particular na fita de uma máquina de Turing. Assim, o número de células usadas cresce consoante a magnitude dos números. O espaço necessário é potencialmente infinito.

Este facto leva-nos a uma conclusão matemática que nos permite caracterizar melhor o problema da paragem:

Teorema 4. A indecidibilidade do problema da paragem deve-se exclusivamente ao facto de o espaço usado durante a computação não poder ser limitado (ou constrangido).

Suponhamos que se sabe *a priori*, em virtude da natureza dos problemas de uma certa classe, que, *para inputs de tamanho n*, não mais de *s(n)* células das fitas das máquinas de Turing são usadas (a Figura 6 ilustra este conceito).

Constrangimentos de tempo implicam constrangimentos de espaço, pois em *t* transições de uma máquina de Turing, não mais de *k(t+1)* células podem ser visitadas, onde *k* é o número de fitas da máquina. O recíproco também é verdadeiro: se uma máquina de Turing repete a mesma configuração no decurso de uma computação, então essa computação é infinita e a máquina não parará.

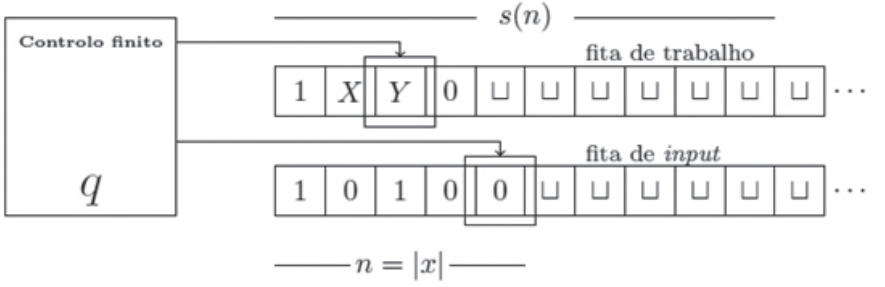

FIGURA 6. Representação de uma máquina de Turing, nomeadamente do controlo finito e das fitas. O espaço de trabalho (indicado na fita de trabalho) está limitado pela função *s* dependente do tamanho do input. Quer isto dizer: quando o input tem tamanho *n*, sabe-se que não mais de *s(n)* casas da fita de trabalho são usadas.

Assim, para as máquinas de Turing, um limite de espaço impõe um limite de tempo: basta atribuir à máquina um contador que delimite o número de configurações diferentes possíveis.

Teorema 5. Se M_1 é uma máquina de Turing que reconhece o conjunto *A* em espaço limitado (mas que não pare necessariamente para todos os *inputs*)[15], então existe uma máquina de Turing M_2 que decide A (e que, portanto, para para todos os *inputs*).

Sem perda de generalidade vamos considerar que a máquina $M_1 = <Q,...>$ tem duas fitas, uma fita de *input* e uma fita de trabalho. O número de configurações da máquina em espaço *s(n)* é majorado por $\#Q \times a^{s(n)} \times s(n) \times n$ que é um número da ordem $2^{cs(n)}$,[16] para alguma constante *c*[17].

15 Em rigor deve dizer-se que o espaço é computável e que cresce mais depressa do que a função logaritmo.

16 Para designar uma configuração é necessário (a) designar um estado de *Q*, (b) designar o conteúdo da fita, que é uma palavra que se escreve com símbolos de um alfabeto de *a* símbolos, (c) designar a posição da cabeça de leitura/escrita da fita de trabalho e (d) designar a posição da cabeça de leitura da fita de *input*. Para (a) temos $\#Q$ possibilidades, para (b) temos $ax...xa=a^{s(n)}$ conteúdos possíveis, para (c) temos uma de *s(n)* casas possíveis e, finalmene, para (d) temos *n* casas possíveis que é precisamente o tamanho do *input*.

17 Estamos assumir que, por hipótese, $s(n) \geq log(n)$, isto é, que o espaço de que a máquina necessita é superlogarítmico.

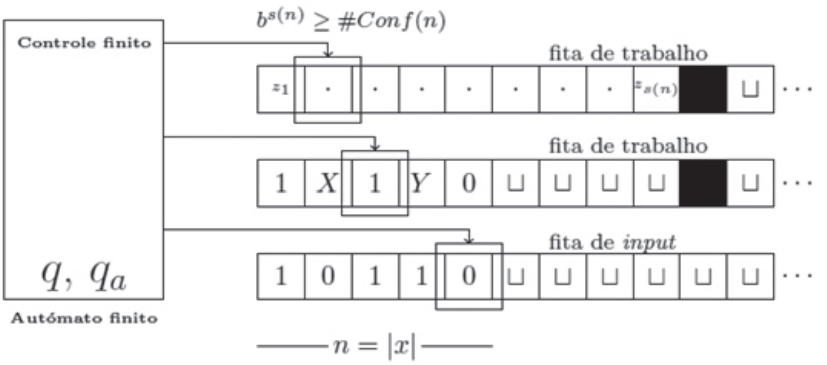

FIGURA 7. Representação de uma máquina de Turing M_2, nomeadamente o controlo finito e as fitas. O espaço disponível *s(n)* está marcado na fita de trabalho. A máquina M_2 faz a contagem, na base *b*, na fita suplementar, do número das possíveis configurações, a saber *#Conf(n)*.

Então, M_2, a máquina de três fitas representada na Figura 7, com uma fita de *input* e duas fitas de trabalho, reserva *s(n)* células nas segunda e terceira fitas[18], simula M_1 na segunda fita, enquanto, na terceira fita, conta, na base $b=2^c$, o número de passos simulados. Quando a contagem chega ao fim, i.e., quando o contador preencher todas as *s(n)* células com o número $b-1$[19], M_2 *sabe que* M_1 está num ciclo infinito e para a simulação no estado de rejeição. Se, antes da contagem chegar ao fim, M_1 atingir uma configuração de paragem, então a máquina M_2 aceita ou rejeita de acordo com M_1.

Deixa-se, como exercício, a especificação informal de uma máquina de Turing *M* que, dadas uma máquina de Turing *N* que opera em espaço finito (embora desconhecido) e um *input w* para *N*, aceita <*N,w*> se *N* para para *w* e rejeita <*N,w*> caso contrário. Esta construção não exige qualquer uso explícito de matemática; é pura e simplesmente um inteligente truque de programação, que restringe o problema da paragem ao caso em que a limitação de memória não pode ser diagnosticada *a priori*.

18 E pode fazê-lo pois a função s é computável.

19 Que é o último possível, pois, na base b, os dígitos são 0, 1, 2, ..., b−1.

Desta maneira, podem resolver-se todos os problemas de decisão que correm em espaço finito, entendendo-se por espaço finito que o espaço necessário para processar cada *input* é finito.

> Mais, mesmo no caso em que alargamos a classe das máquinas de Turing àquelas cujos espaços de trabalho são simplesmente finitos, embora possam variar de input para input, a decisão da paragem tem solução computacional.

O que quer dizer que o problema da paragem está associado ao uso de espaço infinito no processamento de certos *inputs*. Isto é, para haver indecidibilidade é mesmo necessário alargar a classe das máquinas de Turing àquelas que consomem espaço infinito, ou seja, consomem memória infinita.

6. A Máquina Acelerada

O decisão da paragem é um problema da lógica-matemática. O tempo que associamos a uma computação não é o tempo físico, o tempo que marcam os relógios. É antes um tempo lógico. Quando são postos em relação o tempo lógico da máquina de Turing (a simples contagem das transições) e o tempo físico, aprofunda-se a natureza do conceito de computação.

6.1. A eficiência de uma máquina de Turing

A máquina de Turing é um modelo matemático também usado para estudar a eficiência das computações enquanto número de operações efetuadas pela máquina em função do tamanho do *input*. Entenda-se por tamanho do *input* o número de símbolos necessários para o escrever, o que, em termos digitais, corresponde ao número de 0's e 1's necessários para escrever os dados na fita de *input*[20]. Uma eficiência, ou complexidade temporal, de, digamos, n^2 significa que, para *inputs* de tamanho n, não mais de n^2 transições são necessárias desde o estado inicial até ao estado de aceitação ou de rejeição.

20 O tamanho do *input* constitui, assim, uma medida da quantidade de informação disponível nos dados do problema.

Há, obviamente, razões diversas para pretendermos máquinas de Turing eficientes, as quais correspondem a programas de computador igualmente eficientes. Mas a questão que nos traz a esta secção é compreender a relação entre o tempo físico e o tempo lógico que mede a eficiência da máquina. Para este fim, vamos recorrer a um exemplo.

A dinâmica da atmosfera é deveras complexa. E não é difícil compreender porquê...

Em primeiro lugar, a atmosfera é um fluido compressível cuja dinâmica é regida por uma equação da Física-Matemática, a denominada equação de Navier-Stokes: equação muito intrincada do ponto de vista matemático.

Em segundo lugar, a atmosfera deve ser estudada num referencial em rotação, que acompanha a rotação da Terra: este referencial não é um referencial inercial, pelo que há forças de inércia em ação, forças centrífugas e forças ditas de Coriolis, responsáveis pelos ciclones e anticiclones. Em virtude da esfericidade da Terra, as coordenadas naturais não são as coordenadas cartesianas com que todos estamos familiarizados. A equação que descreve a dinâmica da atmosfera, torna-se subitamente muito complexa.

Depois, a atmosfera interage com o oceano, nomeadamente com as correntes oceânicas, e com o relevo da Terra.

Se quiséssemos representar toda a complexidade inerente à descrição do estado da atmosfera, então a equação tornar-se-ia intratável e matematicamente ilegível.

Supondo, o que é de facto impossível, que seríamos capazes de controlar todos os termos da equação da dinâmica da atmosfera, surge-nos um problema inteiramente novo: como resolver a equação? Tal equação *não se resolve pela analogia* – construindo um sistema, no laboratório, análogo à atmosfera; resolve-se por processos numéricos, isto é, partindo do estado da atmosfera (distribuição espacial da pressão, da temperatura, da velocidade do vento, etc.), utilizam-se métodos numéricos (e um supercomputador) para calcular o estado da atmosfera num instante posterior.

Portanto, o problema da dinâmica da atmosfera é, antes de mais, um problema da Física-Matemática e da Análise Numérica: encontrar as equações que descrevem a circulação da atmosfera em pequena e larga escalas e, depois, resolver as equações com base nos dados disponíveis, o que se faz recorrendo a um computador.

Porém, os computadores levam o seu tempo a resolver os problemas da Física-Matemática...

Por exemplo, espera-se que a previsão do tempo para amanhã possa ser obtida ainda hoje... e nunca depois de amanhã.

O computador não simula os fenómenos da natureza no seu próprio tempo, isto é, no tempo físico, a simulação de um processo complexo pode levar muitíssimo mais tempo do que o desenrolar do fenómeno em si. Para poder prever o tempo para amanhã, com elevado grau de confiança, usam-se supercomputadores capazes de realizar muitas operações matemáticas *por unidade de tempo físico*[21].

Exemplos como este evidenciam a maior ou menor capacidade que o computador tem de acelerar *as computações da natureza*, estabelecendo uma relação elementar entre *operação computacional* – transição de uma máquina de Turing – e *unidade de tempo físico*. Acrescente-se, a este respeito, que nem sempre é possível simular um processo físico, tal como a dinâmica da atmosfera, no seu próprio tempo; pode ser necessário muito mais tempo (físico) computacional (em virtude do elevado número de operações a efetuar) do que o tempo do processo físico em si. No caso da atmosfera, o que o computador faz é uma simulação aproximada da evolução do estado do tempo. Também, por isso, as previsões do estado do tempo têm um grau de fiabilidade muito variável, dependendo da estabilidade da atmosfera.

Podemos pensar o contrário. Para resolver certo problema computacional, talvez exista um sistema natural capaz de acelerar o computador eletrónico para além do que possamos imaginar.

21 O computador eletrónico executa algoritmos em tempo físico, o tempo dos relógios. Porém, estamos aqui interessados no número de operações elementares que o computador pode realizar por unidade de tempo físico e, nesta perspetiva, o computador comum assemelha-se a uma máquina de Turing.

6.2. Aceleração

De facto...

Podíamos imaginar computadores capazes de acelerar indefinidamente o processo computacional através da capacidade ilimitada de reduzir o tempo necessário a cada transição (operação). Esta aceleração é meramente conceptual, pois há limites físicos para a aceleração das computações. Para que serve, então, esta experiência do pensamento? É, essencialmente, uma experência que permite caracterizar a natureza de problemas cuja resolução escapa aos computadores (tal como os conhecemos).

Sabemos, assim, através de uma teoria matemática muito bem elaborada, que nem todas as decisões têm solução computacional, *mesmo se tivéssemos a capacidade de reduzir ilimitadamente o tempo de cada transição (operação)*. Quando Turing introduziu a máquina do seu nome, não estabeleceu qualquer equação que envolvesse ambos os conceitos de tempo. O tempo da máquina de Turing é um tempo lógico que carece de qualquer análise física do tipo:

$$\text{n transições} \;=\; \text{t(n) segundos}$$

Sendo assim, nada obsta à experiência conceptual seguinte: imaginemos um engenho físico capaz de simular máquinas de Turing de tal modo que dispende 1 segundo na primeira transição, 1/2 do segundo na segunda transição, 1/4 do segundo na terceira transição e assim sucessivamente, de modo a que, na n-ésima transição, gasta a fração $1/2^{n-1}$ do segundo. Ao cabo de *2* segundos todas as computações estão terminadas,

$$1+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\cdots$$

não importando se são finitas ou infinitas. Este simulador hipotético é mais poderoso do que as máquinas de Turing *per se*, mas não pode resolver todos os problemas de decisão interessantes da Matemática. Pode, no entanto, resolver o problema da paragem. Chama-se a este paradigma computacional conceptual *máquina de Turing acelerada* e foi

introduzido por Jack Copeland (*vide* Copeland, 1998 e Copeland, Super Turing-machines, 1998). A equação dos tempos para a máquina de Turing acelerada é, para *n>1*, dada por:

$$\text{n transições} = 2 - \frac{1}{2^{n-1}} \text{ segundos}$$

Vejamos como uma máquina de Turing acelerada pode resolver o problema da paragem.

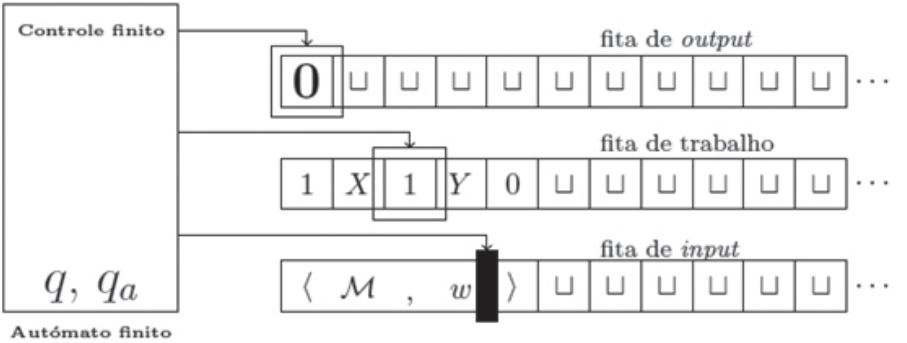

FIGURA 8. Representação de uma máquina de Turing acelerada. No início da computação, a fita de input regista o código *<M,w>* e a fita de *output* regista *0*. A cabeça de leitura da fita de *input* está a varrer o *input <M,w>* que pode ser mais ou menos vasto.

A convenção de *input/output* é a seguinte (tal como consta na Figura 8): no início, o utilizador coloca na fita de *input* o código de uma máquina de Turing, *M*, bem como o *input w* para *M*, o que representamos como o *input* combinado *<M,w>*, conjuntamente com o valor *0* na fita de *output* que denota que a máquina *M* não para para o *input w*. Assim, o utilizador começa por assumir que a máquina *M* não para para o *input w*. Depois, a máquina de Turing acelerada mais não faz do que descodificar o *input* para simular a máquina de Turing *M* com *input w*, o que demora não mais do que *2* segundos. Se a máquina de Turing *M* para para o input *w*, então a máquina acelerada escreve *1* na fita de *output*, no lugar onde estava o *0* e desliga-se. Ao fim dos *2* segundos o utilizador observa a fita de *output* e lê o resultado: se lá estiver 0 é porque a má-

quina *M* não parou para o *input w*; mas, se lá estiver *1*, então o utilizador sabe que a máquina *M* parou para o *input w*. E o problema da paragem das máquinas de Turing ficou decidido em *2* segundos. Pronto!

Bem, nada pronto, pois análises diversas da máquina de Turing acelerada conduziram os cientistas à tese realista de que a máquina explode antes de concluir a computação infinita de *M*, nomeadamente no caso de *M* não parar. Este estralhaçar do computador acelerado resulta das suas necessidades energéticas, à medida que a computação acelerada prossegue.

O mais curioso desta máquina acelerada é que, mesmo assumindo que poderia ser construída, ela carece de suporte conceptual pleno: essencialmente, não pode dizer-se que a máquina de Turing acelerada tenha configuração ao fim de *2* segundos. Pode calcular-se a configuração da máquina no instante $1-1/2^n$, para todo o valor de *n* (*0*, *1*, *2*, *3*, ...), mas não pode dizer-se qual é a configuração da máquina exatamente no instante *t=2* segundos. Mais, a máquina acelerada resolve exatamente o mesmo problema que a máquina de Turing *M*, ao simulá-la: se *M* para para o *input w*, a máquina acelerada dá resultado *1*, mas se a máquina *M* não para para o *input w*, então a máquina acelerada não dá, de facto, solução, pois a solução 0 foi nela instalada no início.

O problema, considerado pelos filósofos da ciência, tais com Oron Shagrir (*vide* o seu relativamente recente artigo (Shagrir, 2012)), prendem-se com *a incapacidade de a máquina acelerada transitar para o comum estado de aceitação*, isto é, não se consegue definir a máquina acelerada de modo a que, aos *2* segundos de funcionamento, a máquina se encontre no estado de aceitação ou de rejeição, nomeadamente no caso das computações infinitas.

7. O Problema dos N Corpos

7.1. As singularidades sem colisões

Não acredito que o problema da paragem, computacionalmente insolúvel, possa ser resolvido por algum sistema físico, isto é um sistema (de natureza mecânica, termodinâmica, eletromagnética, quântica, ou outra) que, dadas as condições iniciais que, de algum modo, codifiquem

uma máquina de Turing com *input*, *<M,w>*, possa, na sua evolução em tempo finito, determinar se a máquina *M* para ou não para o *input w*. O computador digital – digamos o *laptop* – é um sistema físico tal que o *input* para um programa pode ser visto como parte das condições iniciais (digitais) para a resolução do problema.

Um sistema físico que resolvesse o problema da paragem teria o seguinte sabor: lança-se para o espaço um corpo com condições iniciais tais que refletem o código de uma máquina de Turing *M* com *input w*, *<M,w>* : (a) caso o corpo seja capturado pelo planeta Marte, então é porque a máquina de Turing *M* para para o *input w*, caso o corpo seja expelido do Sistema Solar, então é porque a máquina de Turing M não para para o *input w*. Não acredito que tal problema possa ter solução na Física hodierna.

Porém..., se considerarmos sistemas físicos abstratos, corpos abstratos (e.g., sistemas de pontos materiais, sem dimensões, mas providos de massa) no espaço físico abstrato (e.g., tridimensional cartesiano infinito), sujeitos a leis físicas abstratas não necessariamente precisas no contexto da Física Moderna (e.g., a lei da gravitação universal de Newton, que pode ser vista como uma lei assintótica no quadro da Teoria da Relatividade Geral), então *o problema da paragem pode ser resolvido*.

Curioso não é?

Uma solução física para o problema da paragem foi encontrada por Warren Smith (*vide* Smith, 2006) que recorreu a uma das soluções de um problema histórico conhecido por Conjetura de Painlevé.

Quando duas partículas com massa se aproximam, sujeitas à força da sua gravidade recíproca, a sua velocidade aumenta e pode dar-se uma colisão. Nesse momento a distância entre as partículas é zero e a força de atração é infinita. Diz-se tratar-se de uma singularidade de acordo com a definição:

Definição 8. (Singularidade) Uma singularidade é um valor do tempo $t = t^*$ onde a solução da equação física que dá a trajetória de um ponto material deixa de existir.

Sabemos que, no contexto da teoria da gravitação de Newton, para haver uma singularidade, (a) têm de existir pelo menos dois corpos, digamos *i* e *j*, e (b) é necessário que a distância $r_{ij}(t)$, entre os corpos *i* e *j*, se torne arbitrariamente pequena quando o tempo *t* se aproxima arbitrariamente de t^* [22].

E.g., a colisão é uma singularidade. Mas serão colisões todas as singularidades? Este problema foi levantado na viragem do século XIX para o século XX por Painlevé e Zeipel. Painlevé conjeturou que havia singularidades sem colisões (Conjetura de Painlevé). Como é tal possível? Da seguinte maneira: quando o tempo atinge o valor singular t^*, todos os corpos envolvidos[23] já estão no infinito. Eis a singularidade. Parece impossível, mas não é. É uma patologia da física newtoniana.

Soluções uni-, bi- e tridimensionais deste problema foram dadas em diversos momentos históricos ao longo de mais de um século, por cientistas como Sundman, Wintner, McGehee, Gerver, Saari e Xia. Jeff Xia encontrou, na sua tese de doutoramento, uma solução tridimensional muito difícil de 5 corpos (Xia publicou a construção dos 5 corpos no artigo (Xia, 1992)).

A construção de Xia é feita com base em dois sistemas binários (e.g., estrelas duplas) muito excêntricos cujas trajetórias são realizadas em sentidos opostos. Um sistema binário de estrelas é tal como se mostra na Figura 9. Dois destes sistemas colocados axialmente, como na Figura 10, estão suficientemente separados. Um quinto corpo move-se ao longo do eixo vertical e oscila entre os dois binários com frequência crescente. Ao fim de um certo tempo (finito), os cinco corpos estão no infinito: o asteróide oscilante foi ejetado para o infinito e as órbitas excêntricas dos binários, que se vão tornando cada vez mais excêntricas, abrem-se em parábolas com os respetivos corpos também ejetados para o infinito.

22 A recordar: a atração gravitacional entre os corpos *i* e *j*, de massas m_i e m_j, respetivamente, é dada, em valor absoluto, por

$$F = G\frac{m_i\, m_j}{r_{ij}^2}$$

onde *G* é a constante de gravitação universal. Esta força é infinita quando a distância entre os corpos é zero.

23 De facto são pontos materiais sem dimensões, mas a tradição impõe a designação de teoria dos *N* corpos a esta parte da Mecânica Racional.

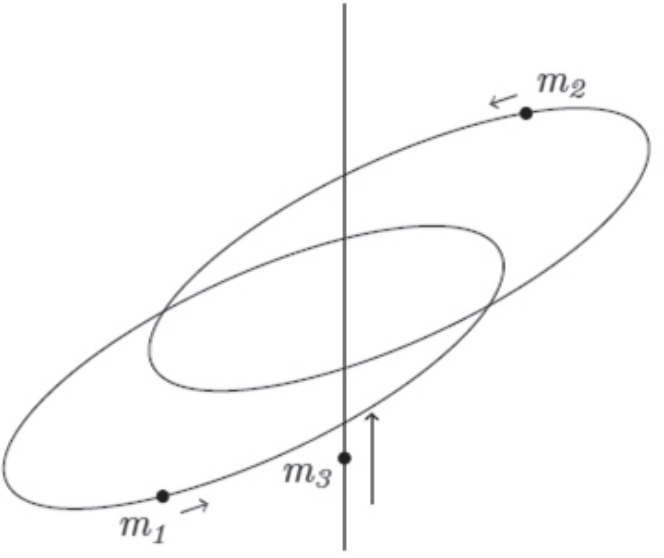

FIGURA 9. Uma estrela binária.

A dinâmica de uma oscilação é mais ou menos esta: o asteróide de massa m_3 passa no primeiro binário quando as duas estrelas estão muito perto uma da outra, extrai energia ao sistema, cujas órbitas se tornam mais excêntricas, e é projetado com grande velocidade em direção ao binário 2 que, nesse momento, tem as duas estrelas também muito próximas uma da outra. O binário 2 detém o asteróide que retrocede projetado em direção ao binário 1 com velocidade ainda maior.

As oscilações do asteróide são progressivamente maiores e mais frequentes, de modo a que, quando $t \to t^*$, a distância que o asteróide percorre em cada oscilação e a frequência das oscilações, bem como os semi-eixos maiores das órbitas das estrelas duplas, tendem para infinito, à medida que os semi-eixos menores das órbitas tendem para zero! O sistema expande-se para o infinito.

Pois bem, existem condições iniciais de massas m_1, m_3 e m_4 (satisfazendo $m_1 = m_2$ e $m_4 = m_5$, m_3 maior do que m_1 e m_4), posições e velocidades iniciais tais que, num tempo finito (!), todos os cinco corpos estão no infinito!

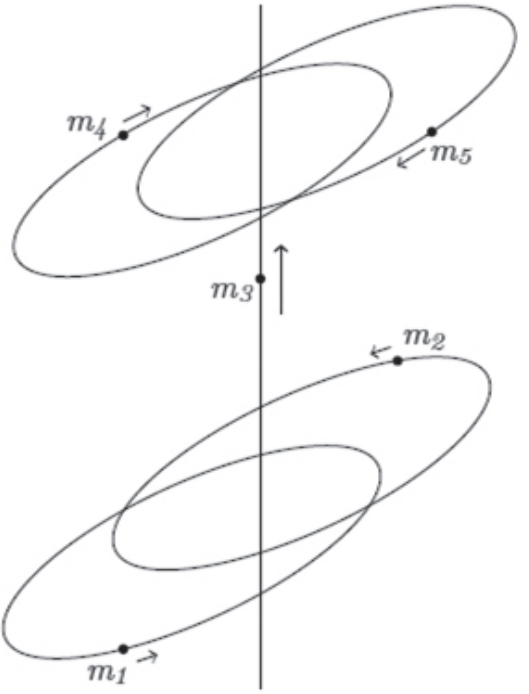

FIGURA 10. Duas estrelas binárias.

A solução de Xia corresponde a 100 anos de história da mecânica celeste e constitui uma conquista intelectual notável.

Como é possível? Sempre que o asteróide passa nas proximidades de um dos binários, extrai-lhe energia e aumenta a sua velocidade. Isto paga-se reduzindo os semi-eixos menores das órbitas dos binários. As condições podem ser de tal forma calculadas que o quinto corpo oscila entre os dois binários, atingindo em cada oscilação velocidades maiores do que na precedente; em ambos os binários, as estrelas passam cada vez mais próximas uma da outra, mas as órbitas são elipses cada vez mais excêntricas; os semi-eixos maiores das elipses são cada vez maiores e os semi-eixos menores cada vez menores.

Em tempo FINITO, os cinco corpos estão no infinito. Diz-se uma singularidade sem colisão (de facto, pode dizer-se, as colisões acontecem no infinito).

7.2. O problema da paragem

Uma máquina de Turing tem um certo número (finito) de estados, digamos *s* (não incluindo os estados de aceitação e de rejeição), e um certo número finito de símbolos no seu alfabeto de trabalho, digamos *l*. O matemático Claude Shannon demonstrou em 1956 [24] que, para toda a máquina de Turing *M* de *s* estados e *l* símbolos, existe uma máquina de Turing *N* de apenas *2* estados (incluindo o estado inicial, mas não contando com os estados de aceitação e rejeição), e muitos mais símbolos, digamos $4s + l$ símbolos, que realiza a mesma função, isto é, que para todo o *input* produz o mesmo *output* que *M*. As máquinas *M* e *N* são pois equivalentes.

Assim, podemos mesmo, em termos abstratos (e, consequentemente, não muito práticos), trabalhar com máquinas de Turing de *2* estados relevantes, digamos o estado inicial q_0 e um estado q_1 (os estados de aceitação e rejeição são estados em que a máquina se desliga e, consequentemente, só podem ocorrer uma vez em cada computação).

Consideremos, pois, todas as máquinas de Turing de 2 estados, q_0 e q_1 (e um número arbitrário de símbolos). Elas esgotam tudo o que é decidível, tudo o que é computável. Independentemente do que cada máquina de Turing faz, podemos enriquecer a sua atividade da seguinte maneira: acrescentamos ao controlo finito de cada máquina novas transições tais que, sempre que a máquina transita para o estado q_0 imprime 0 na fita de *output* e sempre que a máquina transita para o estado q_1, imprime 1 na fita de *output*.

Desta maneira a máquina de Turing, à medida que realiza a sua computação, vai imprimindo uma sequência de *0*'s e de *1*'s. Se a máquina não para, então imprime uma sequência interminável de *0*'s e de *1*'s; caso a máquina pare, a sequência impressa é finita.

Agora consideremos um "binário" de dois corpos, orbitando cada um em torno do outro, como nas Figuras 11 e 12. É constituído por duas "estrelas" B e C à mesma distância do centro de massa do sistema. Na Figura 11, as estrelas B e C estão representadas em dois instantes de tempo, *antes* e *depois*.

24 Num famoso artigo de um livro editado por ele e John McCarthy (Shannon, 1956).

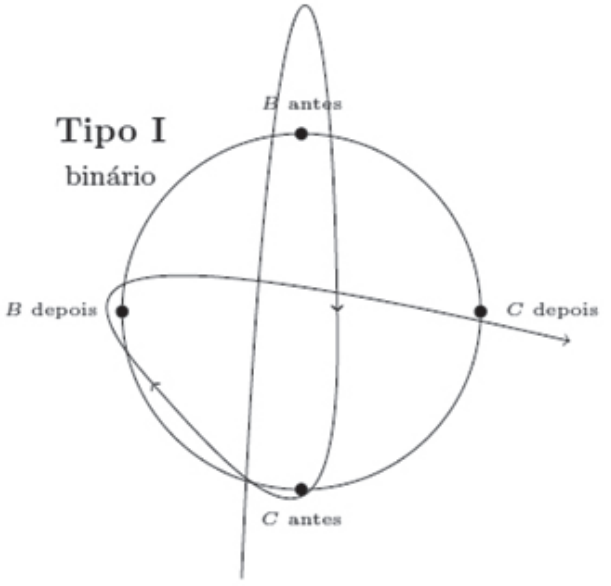

FIGURA 11. Trajetória de Tipo I.

Um asteróide orbita ambas as estrelas em duas trajetórias possíveis: (a) Na trajetória da Figura 11, o asteróide orbita por fora das estrelas B e C e sai do sistema binário contornando, de novo, as estrelas B e C – é a trajetória *0*, isto é, diz-se de Tipo I e é denotada por *0*; (b) na trajetória da Figura 12, o asteróide ziguezagueia por entre as duas estrelas – é a trajetória 1, isto é, diz-se de Tipo II e é denotada por 1. Escusado será dizer que podem escolher-se massas para as estrelas B e C e asteróide, posições e velocidades iniciais, e semi-eixos maiores e menores dos sistemas binários, de modo a que o asteroide possa, em ambos os casos, traçar tais trajetórias no espaço físico bidimensional. *Mais, os parâmetros podem ser escolhidos de tal maneira que o ângulo entre as velocidades de entrada e saída sejam bem determinados*.

Podemos prosseguir pensando assim: cada máquina de Turing imprime uma sequência de *0*'s e *1*'s; a cada uma destas sequências podemos associar trajetórias de *N* asteróides num sistema de *N* binários de estrelas, dispostos no plano em polígono regular como na Figura 13; se a sequência é *0010* ..., então o asteróide 1 executa uma trajetória Tipo I

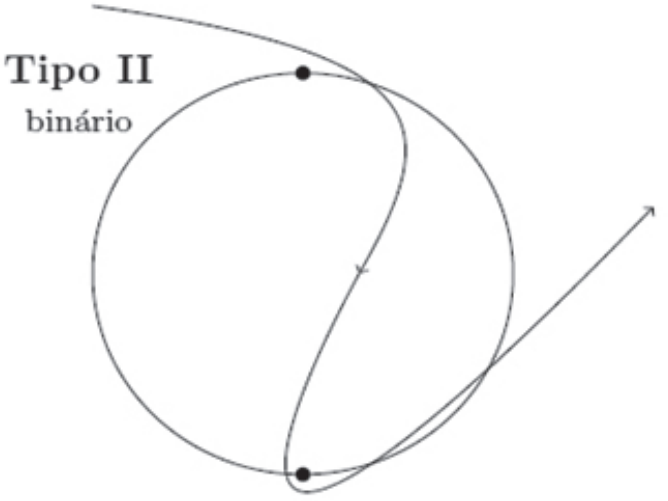

FIGURA 12. Trajetória de Tipo II.

no binário 1 (o que corresponde ao primeiro *0*), sai em direção ao binário 2, executa uma trajetória Tipo I no binário 2 (o que corresponde ao segundo *0*), sai em direção ao binário 3, executa uma trajetória Tipo II no binário 3 (o que corresponde ao primeiro *1*), é expelido em direção ao binário 4, onde executa uma trajetória Tipo I, etc. Cada um dos binários mais asteróide é espelho de outro binário mais asteróide.

Quando um dos asteróides volta ao binário 1, a sua trajetória não se repete necessariamente (não tem de ser cíclica)! Desta vez pode executar um novo troço de Tipo I, mesmo que, da última vez, tenha executado um troço de Tipo II; e, vice-versa, pode executar novo troço de Tipo II, mesmo que, da última vez, tenha executado um troço de Tipo I.

Facto curioso, existem massas, condições iniciais para os *2N+N* corpos (*2N* estrelas e *N* asteróides) de modo a que o asteróide é expelido de cada binário na direção correta e no instante correto de modo a intercetar o binário seguinte nas condições desejadas.

Esta solução bidimensional da singularidade sem colisões é uma variante da construção de Joseph Gerver (o artigo de Gerver é citado nas referências em (Gerver, 1991)). A solução é possível para um número suficientemente elevado de binários, mais os asteróides que circum-

navegam os binários da maneira descrita. *Há soluções de trajetórias infinitas em, digamos, 1 segundo, as quais correspondem à ejeção de toda a massa para o infinito, acompanhada de um número infinito de circum-navegações*. O tempo de *1* segundo é retórica, pois a construção pode ser feita para qualquer intervalo de tempo finito. Em cada circum-navegação, o polígono expande-se, as velocidades dos corpos são modificadas, mas, no caso da construção de Gerver, bem como nesta variante, as formas das órbitas dos binários permanecem essencialmente as mesmas. No caso de singularidade, os corpos estão no infinito. Há também trajetórias finitas que correspondem a uma implosão do sistema de corpos no seu centro de massa, no centro do *N*-gono em *1* segundo. Este sistema discrimina, portanto, as trajetórias finitas das trajetórias infinitas em *1* segundo: as primeiras correspondem a uma implosão, as segundas a uma explosão. Quer isto dizer que um programador sublime, tendo colocado os *2N+N* corpos nas suas posições e com as velocidades

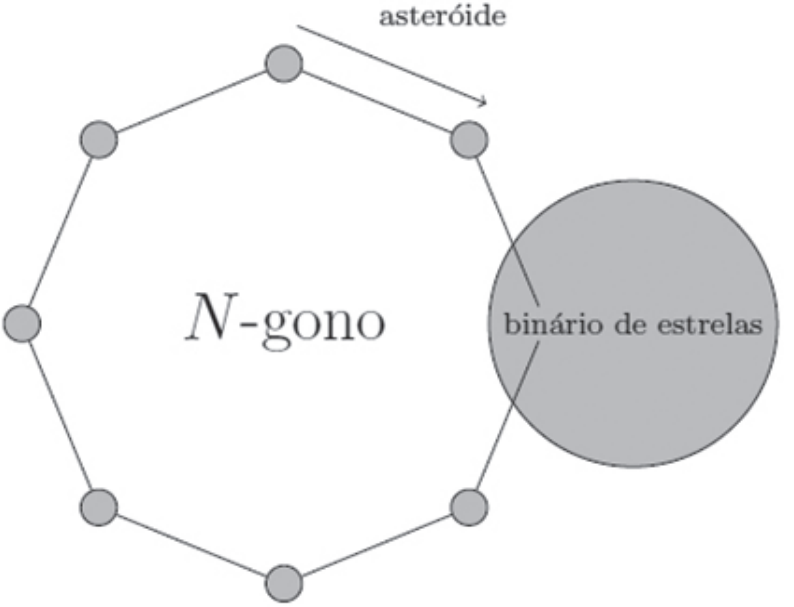

FIGURA 13. Polígono cujos vértices são estrelas binárias afastadas umas das outras, de modo a que os efeitos da gravitação de umas em relação às outras sejam desprezáveis. Os asteróides são projetados de binário para binário. Em cada binário, cada asteróide descreve uma trajetória de Tipo I ou de Tipo II, representadas nas Figuras 11 e 12.

iniciais subtilmente escolhidas (o *input*), apenas tem de olhar para o sistema, digamos *2* segundos depois, para ler o *output*, isto é ou o sistema implodiu ou explodiu. Como vamos elucidar a seguir, este *computador* dos *N* corpos de Warren-Gerver subsume a solução do problema da paragem das máquinas de Turing.

A sequência de *0*'s e *1*'s que é produzida pela máquina de Turing (designada *topologia*) pode ser usada como *input* para uma segunda máquina de Turing M_2 que, a partir da topologia[25] calcula as condições iniciais dos *2N+N* corpos de modo a produzir essa mesma sequência de trajetórias atómicas, ou troços, nos vários sistemas binários que os asteróides vão encontrando. Essencialmente, essa segunda máquina de Turing emprega métodos numéricos para, dada a trajetória global, ou um prefixo desta, em termos de tipos de troços, obter as condições iniciais do sistema. Diz-se que tal máquina de Turing resolve o problema inverso (o problema direto seria o de encontrar a trajetória dadas as condições iniciais). De facto, esta segunda máquina de Turing M_2 tira partido do facto de que, para condições iniciais numa certa vizinhança da condição inicial procurada, a trajetória é essencialmente a mesma. Assim, a segunda máquina lê prefixos de uma trajetória global e calcula as respetivas condições iniciais com a precisão (finita) desejada, de modo a que cada prefixo possa ser calculado a partir das condições iniciais.

Uma terceira máquina de Turing M_3 parte das condições iniciais computadas pela máquina de Turing M_2 para simular as trajetórias dos *2N+N* corpos em *1* segundo. A Figura 14 mostra o encadeamento da computação.

A máquina M_3 pode ser descrita da seguinte maneira: dada como *input* a condição inicial do sistema, de modo a que a máquina M_3 possa obter dela um número de algarismos significativos progressivamente maior, se deles necessitar, M_3 simula o movimento do sistema de *2N+N* corpos com precisão suficiente para decidir sobre a topologia das trajetórias dos corpos em 1 segundo. Podemos enunciar um teorema:

25 Isto é, a partir da sequência dos tipos de circum-navegações: 0 corresponde a um troço de trajetória de Tipo I e 1 corresponde a um troço de trajetória de Tipo II.

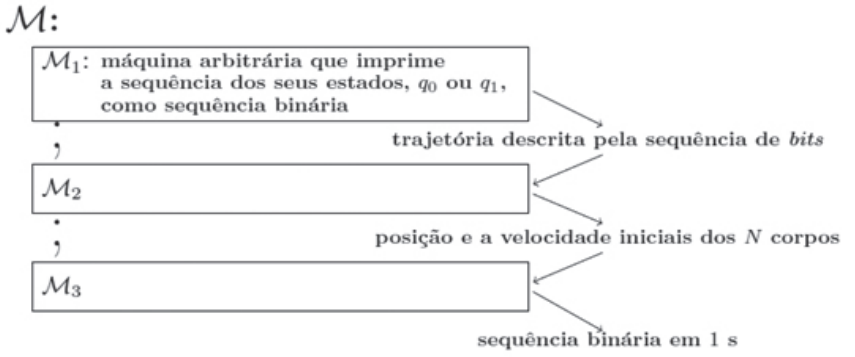

FIGURA 14. Composição *M* de três máquinas de Turing: M_1 imprime a sequência binária, possivelmente infinita, dos seus dois estados, M_2 vai calculando, a partir dos prefixos da trajetória, uma sucessão convergente de condições iniciais e M_3 simula o sistema de *2N+N* corpos a partir das condições iniciais calculadas pela máquina M_2.

Teorema 6. A máquina global, que resulta do processamento paralelo das máquinas M_1, M_2 e M_3, designada por $M = M_1||M_2||M_3$ (*vide* Figura 14), para se e só se os *2N+N* corpos não atingem a singularidade em 1 segundo.

O que é crucial nesta discussão é entender que a máquina M_3 está apenas a resolver equações para determinar o comportamento dos *2N+N* corpos em *1* segundo, podendo garantir-se que, antes da primeira singularidade de qualquer sistema de corpos, a solução existe, é única, bem comportada e pode ser aproximada, tanto quanto se quiser, por métodos numéricos, requerendo-se precisão ilimitada (que os computadores não podem dar, devido à sua limitação de memória, mas que as máquinas de Turing podem), podendo para isso levar tempo infinito.[26]

Podem demonstrar-se os teoremas seguintes:

Teorema 7. *N* pontos materiais num plano, cujas massas, posições e velocidades iniciais se encontram num hipercubo de números reais[27], podem descrever um número não contável de trajetórias distintas em *1* segundo.

26 Diz-se que é analítica.

27 $\mathbb{R}^{5n}$.

As posições e as velocidades iniciais dos corpos de forma a produzir-se uma trajetória de determinado tipo são números reais computáveis.

Um simulador de máquinas de Turing pode apenas imprimir uma trajetória finita de entre um certo número finito de trajetórias finitas, num intervalo de tempo finito, tão grande quanto se queira.

Estes resultados dizem-nos que a possibilidade de simular a trajetória em todos os casos é, pura e simplesmente, impossível! A máquina de Turing (o computador) parte de condições iniciais (*input*) necessariamente computáveis (e, portanto, em número contável), mas o número de trajetórias possíveis em *1* segundo é não contável. Mais, as condições iniciais expressas por números não racionais (mas computáveis como sequências de dígitos), obrigam a máquina a calcular os dígitos um a m, à medida que mais e mais precisão é necessária, de modo a que as máquinas M_2 e M_3 vão demorando mais e mais tempo (transições) para obterem a precisão desejada, num ciclo infinito para topologias infinitas em *1* segundo.

A simulação é feita por aproximações sucessivas (Figura 14): (a) a máquina M_1 vai produzindo prefixos da topologia da trajetória; (b) a máquina M_2 toma como *input* um prefixo de cada vez, calculando, pelo método inverso, as condições iniciais do sistema de modo a originar essa topologia[28]; finalmente, a máquina M_3, para cada aproximação às condições iniciais, faz a simulação da dinâmica do sistema em *1* segundo. As aproximações sucessivas são concluídas em tempo finito, todas elas. Por cada prefixo de uma trajetória possivelmente infinita, calculam-se as respetivas condições iniciais, a partir das quais se faz a simulação para *1* segundo. Se a máquina M_1 para, então, para o prefixo final, coincidente com a trajetória completa, que deverá ser concluída com implosão do sistema, a máquina M_2 acabará por obter as condições iniciais suficientemente precisas para que a máquina M_3 possa realizar a simulação com êxito. Mas, se a máquina M_1 não para, então, para cada prefixo da topologia em *1* segundo, a máquina M_2 procede ao cálculo das condições iniciais que conduzem à implosão, após essa trajetória

28 O cálculo das condições iniciais é, no artigo original de Warren Smith, feito pelo designado método de Runge-Kutta. O objetivo deste método é poder obter-se o resultado, as condições iniciais com precisão ilimitada.

do asteróide, e a máquina M_3 realiza a respetiva simulação. Claro está que, neste caso, a trajetória no intervalo de um segundo vai ficando progressivamente maior e, no limite, é infinita e, consequentemente, ao invés de uma implosão o sistema atinge a singularidade, sendo ejetado para o infinito.

Vemos assim que, enquanto a máquina de Turing M_1 não para, *o sistema físico dos 2N+N corpos resolve o problema em 1 segundo*. Quer dizer, o sistema físico dos *2N+N* corpos resolve em 1 segundo o problema da paragem da máquina de Turing M_1. Como toda a máquina de Turing se acha representada entre as máquinas de *2* estados, concluímos que existe um sistema físico, a dinâmica de *2N+N* corpos no espaço cartesiano bidimensional, sujeitos à gravitação newtoniana, que constitui um computador abstrato, um solucionador do problema da paragem das máquinas de Turing.

Sendo assim, podemos concluir que, para cada máquina de Turing *M*, existem condições iniciais para um sistema de *2N+N* corpos, tais que, em *1* segundo, o sistema atinge a singularidade se e só se *M* não para. Nesta perspetiva, a dinâmica dos *2N+N* corpos, com certas condições iniciais, representa a máquina de Turing *M*. A mecânica newtoniana proporciona a solução em *1* segundo de tempo físico, o qual encapsula uma infinidade de transições de *M*. De facto, cada transição da máquina de Turing, corresponde, no sistema físico, a um intervalo de tempo cada vez mais pequeno, à semelhança da máquina de Turing acelerada da Secção 6: o problema é resolvido em virtude de extração de energia do campo gravitacional em quantidade ilimitada. O tempo lógico da computação é pago em energia.

Este resultado permite que concluamos que a Física, com generalidade, não é de todo simulável. E.g., não pode simular-se, com toda a generalidade, a dinâmica de *N* corpos num computador. Mais, Warren Smith mostrou também que não pode simular-se, com toda a generalidade, a dinâmica de um fluido num computador. Smith construiu um recipiente com um vértice que, através da dinâmica de um fluido nele contido, sujeito a um regime da equação de Navier-Stokes da Mecânica dos Fluidos, permite resolver o problema da paragem em *1* segundo.

As singularidades desaparecem no quadro de certas versões da Teoria Geral da Relatividade de Einstein, tal como a designada Teoria Geral da Relatividade Linearizada, o que, por um lado, elimina a possibilidade de resolver o problema da paragem através da dinâmica de *N* corpos, por outro lado, torna a Física de novo simulável. Porém, num contexto mais geral, o estudo das singularidades do espaço-tempo – dos já célebres buracos negros – abre de novo as portas à resolução do problema da paragem, mas agora um pouco à semelhança das máquinas de Turing aceleradas; não se mexe nas máquinas em si, mas joga-se com os observadores localizados em regiões diversas do espaço-tempo. Embora muito interessantes, se não mesmo fascinantes, os computadores relativísticos necessitariam de um artigo inteiro para poderem ser rudimentarmente apresentados[29].

Como as simulações não podem ser válidas em todos os casos, concluímos que a Física, tal como a conhecemos, não é simulável com generalidade. Há outros exemplos, mas, por ora, bastam estes[30].

8. Conclusão

Neste artigo estudámos como pode interpretar-se em termos físicos o mais conhecido problema limite da computação: o problema de decidir, dado um programa *M* e um *input w*, se *M* a operar sobre *w* para ou não, ou, dado um programa *M*, se *M* para ou não para todos os possíveis *inputs*.

Vimos que as soluções físicas conhecidas para este problema existem apenas no limite da realidade física das máquinas aceleradas ou dos sistemas de corpos com massa mas sem dimensões, sujeitos à sua gravidade recíproca. Ora as máquinas aceleradas requerem quantidades infinitas de energia num intervalo de tempo finito e os corpos sem dimensões constituem uma abstração útil à Física, mas não realizável.

29 Os artigos (Hogarth, 2004) e (Andréka, Németi , & Németi, 2009) abrem as portas ao estudo dos computadores relativísticos.

30 Roger Penrose, no seu livro (Penrose, 1989), discute alguns exemplos de física não simulável.

De qualquer modo, é uma grande surpresa ver como o problema da paragem está relacionado com a dinâmica de *N* corpos... Por um lado, temos a mecânica newtoniana, onde a interação única é a força de gravitação universal. Por outro lado, temos a natureza das transições da máquina de Turing. Que relação há entre estes dois mundos? Aparentemente nenhuma. A relação encontra-se na virtualidade que os fenómenos naturais, levados ao limite, têm de encerrar os designados processos de Zenão, processos que permitem encapsular uma infinidade de eventos no célebre segundo retórico – as oscilações de um asteróide num intervalo de tempo finito!

Pode pensar-se que estas incomputabilidades da Física acabam com o advento da Física Moderna. Mas não. David Geroch e James Hartle, dois eminentes físicos teóricos, num seu artigo sobre Fundamentos da Física em (Geroch & Hartle, James B., 1986), põem em evidência possíveis fontes de incomputabilidade na física das partículas.

O problema da paragem é apenas um (o mais conhecido) dos incontáveis problemas de decisão que não têm solução computacional. Incontáveis significa que não podem ser contados. Os problemas de decisão que têm solução computacional são contáveis como os números naturais: *0*, *1*, *2*, ... em direção ao infinito. Problemas de decisão incontáveis são como os números reais, infinitos, mas de uma infinidade superior à dos números naturais.

O nosso estudo dos contornos do problema da paragem serve apenas para mostrar que a Física, tal como é equacionada, e estudada pelos físicos e matemáticos, nem sempre tem simulação possível em termos computacionais.

Ora uma Física que careça de instrumentos matemáticos preditivos, uma Física meramente qualitativa, é uma Física que pode ter interesse teórico, mas é, na prática, ineficaz.

Pode mesmo afirmar-se que os limites da computação impõem limites à Física, no sentido em que a Filosofia da Ciência deverá preferir as teorias preditivas às teorias não preditivas.

Referências

Andréka, H., Németi , I., & Németi, P. (2009). General relativistic hypercomputing and foundation of mathematics. *Natural Computing, 8*(3), 499–516.

Copeland, J. B. (1998). Even Turing machines can compute uncomputable functions. In C. Calude, Casti, John, & Dinneen, M. J. (Ed.), *Unconventional Models of Computation, Lecture Notes in Computer Science* (pp. 150–164). Springer.

Copeland, J. B. (1998). Super-Turing machines. *Complexity, 4*, 30–32.

Costa, J. (2012). Turing machines as clocks, rulers and randomizers. *Boletim da Sociedade Portuguesa de Matemática*(67), 121–153.

Davis, M. (2000). *The Universal Computer, The Road from Leibniz to Turing.* W. W. Norton and Company.

Fritz, W. B. (1996). The women of ENIAC. *IEEE Annals of the History of Computing, 18*(3), 13–28.

Geroch, R., & Hartle, James B. (1986). Computability and physical theories. *Foundations of Physics, 16*(6), 533–550.

Gerver, J. (1991). The existence of pseudo-collisions in the plane. *Journal of Differential Equations*(89), pp. 1–68.

Hogarth, M. (2004). Deciding Arithmetic Using SAD Computers. *British Journal for the Philosophy of Science, 55*(4), 681–691.

Penrose, R. (1989). *The Emperor's New Mind.* Oxford University Press.

Shagrir, O. (2012). Supertasks do not increase computational power. *Natural Computing, 11*(1), 51–58.

Shannon, C. E. (1956). A universal Turing machine with two internal states. (C. E. Shannon, & J. McCarthy, Edits.) *Annals of Mathematical Studies, 34*, pp. 157–165.

Smith, W. (2006). Church's thesis meets the N-body problem. *Applied Mathematics and Computation, 178*(1), 154–183.

Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, 42*, 230–265.

Turing, A. (1937). On computable numbers. *Proceedings of the London Mathematical Society, 43*, 544–546.

Turing, A., & Girard, J.-Y. (1991). *La Machine de Turing.* Sources du Savoir, Seuil.

Xia, Z. (1992). The existence of noncollision singularities in Newtonian systems. *The Annals of Mathematics, Second Series, 135*(3), 411–468.