3. Outras abordagens à computabilidade

3.1 Outras abordagens à computabilidade

Nos últimos 70 anos têm sido apresentadas diferentes propostas para uma completa caracterização matemática da noção intuitiva de computabilidade. A abordagem URM é uma das mais recentes.

Algumas das abordagens alternativas são as seguintes:

(a) **Gödel-Herbrand-Kleene** (1936) Funções recursivas gerais definidas por cálculo equacional

(b) **Church** (1936) Cálculo lambda (λ -calculus)

(c) **Gödel-Kleene** (1936) Funções μ -recursivas e funções parciais recursivas

(d) **Turing** (1936) Funções computáveis por máquinas de Turing

(e) **Post** (1943) Funções definidas por sistemas de dedução canónicos

(f) **Markov** (1951) Funções dadas por certos algoritmos em alfabetos finitos

(g) **Shepherdson-Sturgis** (1963) Funções computáveis por URM

3.1 Outras abordagens à computabilidade

Resultado fundamental

Cada uma das anteriores propostas para uma completa caracterização da noção de computabilidade efectiva dá origem à mesma classe de funções , a classe $\mathbf{C}$.

3.2 Funções parciais recursivas (Gödel-Kleene)

Definição

A classe **R** das **funções parciais recursivas** é a menor classe de funções parciais que contém as funções básicas 0, x+1, U_i^n e é fechada para a substituição, recursão e minimização**.**

(Ou seja, R é a classe de funções parciais que podem ser geradas a partir da

básicas por um número finitos de operações de substituição, recu

Note-se que nenhuma restrição é colocada ao uso do operador-m, logo R contém funções não totais.

Teorema: **R = C**

3.3 Funções recursivas primitivas

Definição

A classe **PR** das **funções recursivas primitivas** é a menor classe de funções que contém as funções básicas 0, x+1, U_i^n e é fechada para a substituição e recursão.

Notas:

- **PR** é fechada para a soma, produto e minimização limitadas.
- Nem todas as funções computáveis são recursivas primitivas (cf. *Função de Ackermann*).

Alan Turing

Alan Turing nasceu em 1912 em Londres.

Turing interessou-se por uma das questões levantadas por Russel, que consistia na possibilidade de mecanizar o raciocínio matemático, capturand sistema todos os princípios válidos da matemática. Nesse sistema, as verdades fluiriam mecânicamente como consequências imediatas de um pequeno conjunto de axiomas (como automóveis numa linha de produção).

Quando, em 1931, o austríaco Kurt Gödel demonstrou a impossibilidade de uma perfeita formalização do raciocínio matemático, dada a existência de proposições indecidíveis em qualquer sistema axiomático consistente, Turing tentou construir um sistema que fosse capaz de determinar se uma proposição era indecidível nesse sistema.

Surge então a **Máquina de Turing**.

Alan Turing

A máquina de Turing, considerada como um precursor do computador, foi inventada em 1936, dez anos antes de serem construídos os primeiros computadores. Mais tarde, Turing foi um dos pioneiros do seu desenvolvimento.

Entretanto, durante os anos da 2ª guerra mundial, Turing foi destacado para auxiliar a descodificação das mensagens dos alemães. A sua contr fundamental para a localização dos submarinos germânicos.

No final da guerra tentou conseguir fundos para a construção de inteligente. Nunca os conseguiu.

Morreu com 41 anos, provavelmente por suicídio.

“As máquinas podem pensar ?”

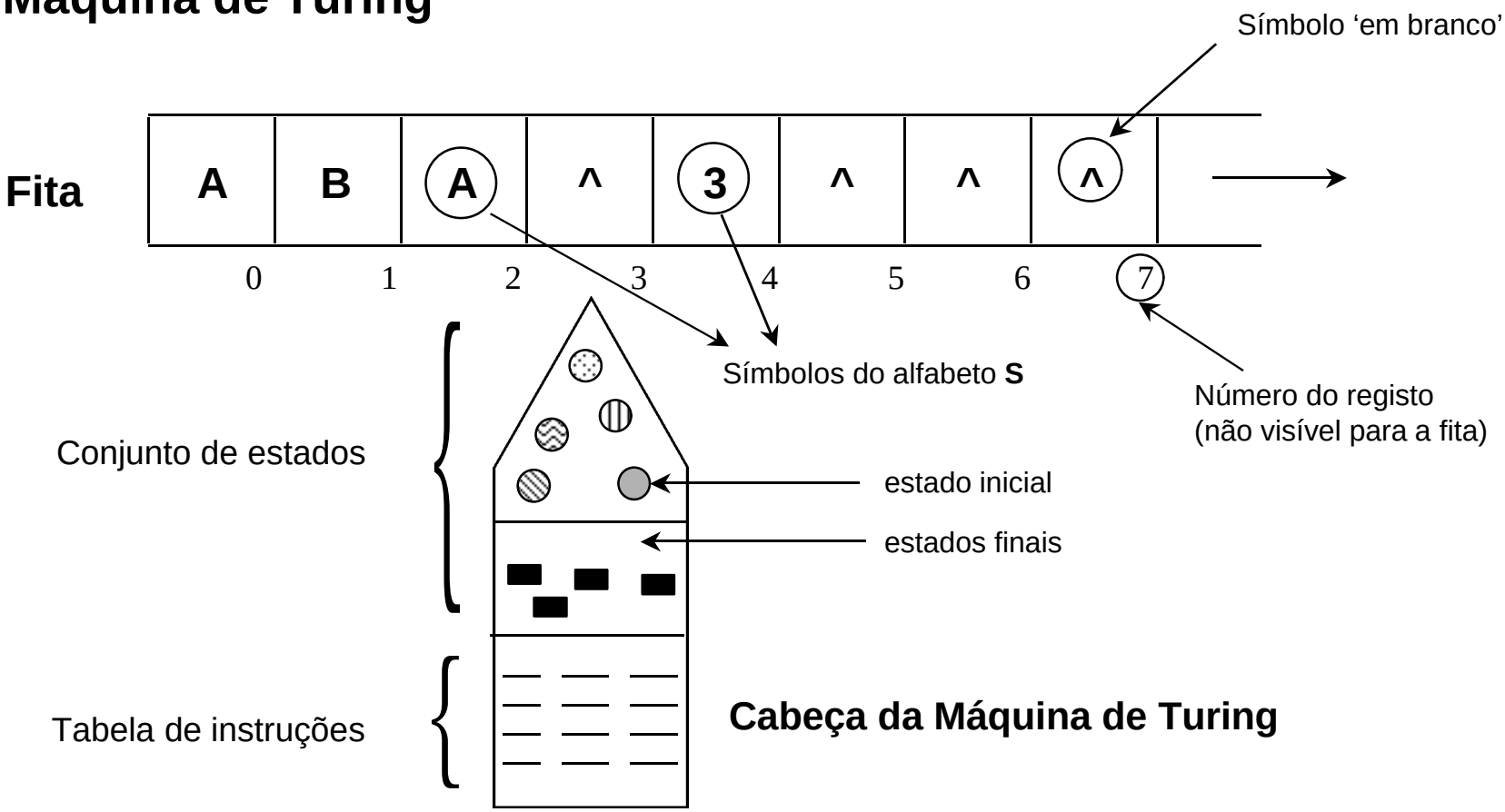
Turing chegou à conclusão que a construção de uma máquina que detectasse (e eliminasse) de um sistema as suas proposições indecidíveis era i confirmando o resultado de Gödel.

Contudo, Turing sempre acreditou na possibilidade de criar uma máquina “inteligente” e num célebre artigo, descreveu um jogo, conhecido agora como o **Teste de Turing**:

“ Três pessoas: um homem (A), uma mulher (B) e o interrogador (C) (pode ser de qualquer sexo). O interrogador encontra-se numa sala, separado dos outros dois. O objectivo do jogo é o interrogador determinar quem é o h a mulher. Ele conhece-os pelas letras X e Y. No final, deve dizer: X é A e Y é B. O interrogador pode colocar questões a ambos, mas desconhece que o objectivo de A é enganá-lo e o de B é ajudá-lo. O que acontece se for um computador a fazer o papel de A ? O interrogador enganar-se-á tantas vezes quantas as que se engana quando o jogo é entre um homem e uma mulher?”

3.4 Computabilidade de Turing

Máquina de Turing



3.4 Computabilidade de Turing

Componentes da Máquina de Turing

Uma máquina de Turing consiste numa fita e numa cabeça que se move para a esquerda e para a direita, lendo ou escrevendo à medida que se move. Em cada momento, a máquina encontra-se num dado estado, de entre um conjunto finito de estados possíveis. Dependendo do estado em que se encontra, uma tabela de instruções comunica-lhe a acção a executar.

A fita

É a memória da MT. É composta por um conjunto infinito de registos, numerados de 0, 1,...

O alfabeto

Em cada registo é escrito um único símbolo, pertencente a um alfabeto finito com **n** símbolos. Este alfabeto tem pelo menos dois símbolos, um dos qu símbolo “em branco” (^), que significa que o registo está vazio.

3.4 Computabilidade de Turing

A cabeça de leitura e escrita

Permite ler e escrever num dado registo da fita.

O conjunto de estados

Em cada momento a máquina encontra-se num determinado estado. O conjunto finito de estados possíveis é descrito por $\{q_1,\ldots, q_m\}$.

A tabela de instruções (Especificação)

A acção a executar em cada momento depende do estado em que a máquina se encontra e do símbolo que está a ser lido. A especifi um conjunto finito de instruções.

3.4 Computabilidade de Turing

Os quádruplos têm a seguinte forma: $\mathbf{q_i\ s_j\ a\ q_l}$, onde $1 \leq i,l \leq m$ e $0 \leq j,k \leq n$

De acordo com cada instrução, amáquina opera do modo seguinte:

(1) Se a máquina se encontra no estado $\mathbf{q_i}$ e o símbolo a ser lido é $\mathbf{s_j}$, então:

Se $\mathbf{a = s_k}$, apaga $\mathbf{s_j}$ e escreve $\mathbf{s_k}$ no registo corrente;

Se **a = R**, move a cabeça para a direita;

Se **a = R**, move a cabeça para a esquerda;

(2) A máquina passa para o estado $\mathbf{q_l}$.

Para cada par $\mathbf{q_i\ s_j}$, deve existir, no máximo, um quádruplo da forma $\mathbf{q_i\ s_j\ a\ q_l}$, para evitar ambiguidades.

3.4 Computabilidade de Turing

Para iniciar a execução de uma MT é necessário que a cabeça esteja colocada na posição inicial e que um estado inicial seja atribuído.

A execução termina quando, num dado estado $\mathbf{q_i}$, ao ser lido o símbolo $\mathbf{s_j}$, não existe nenhum quádruplo da forma $\mathbf{q_i\ s_j\ a\ q_l}$.

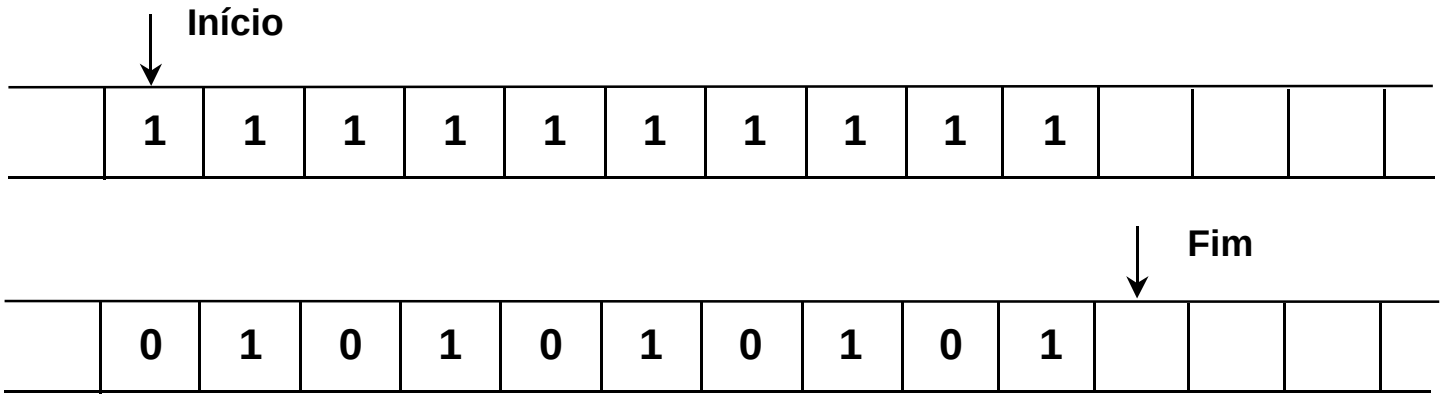
Exemplo: Seja M uma máquina de Turing cujo alfabeto consiste nos símbolos 0, 1 (e o símbolo em branco) e os estados possíveis são $\mathbf{q_1}$ e $\mathbf{q_2}$. A especificação de M é:

$\mathbf{q_1\ 0\ R\ q_1}$

$\mathbf{q_1\ 1\ 0\ q_2}$

$\mathbf{q_2\ 0\ R\ q_2}$

$\mathbf{q_2\ \ 1\ R\ q_1}$



3.4 Computabilidade de Turing

Formas de representação da especificação de uma MT

Relativamente ao exemplo anterior, podemos escrever o conjunto de instruções da máquina de formas distintas e equivalentes:

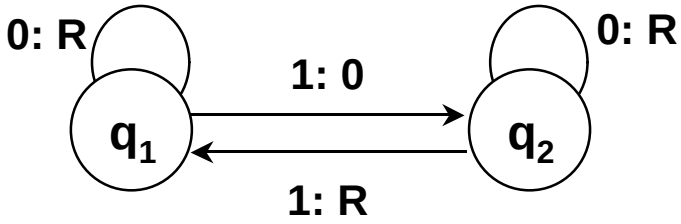
Lista de quádruplos

q_1 0 R q_1
q_1 1 0 q_2
q_2 0 R q_2
q_2 1 R q_1

Tabela

		Símbolo lido	
		0	1
Estado corrente	q1	R q1	0 q2
	q2	R q2	R q1

Grafo



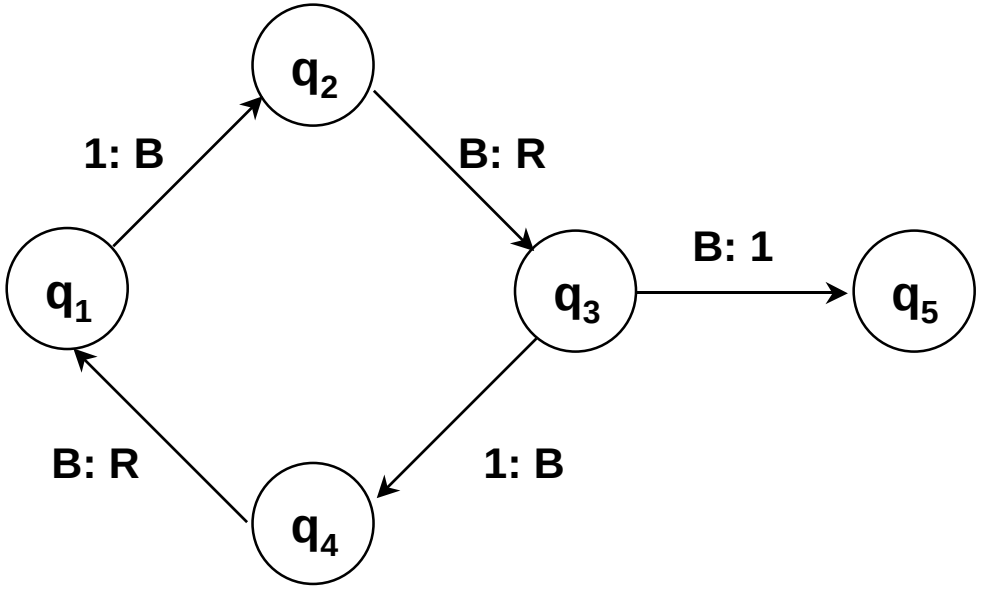
3.4 Computabilidade de Turing

Exercício

O que faz a seguinte máquina de Turing?

O alfabeto é {1, B}, onde B é o símbolo “em branco”. A máquina c
símbolo mais à esquerda e a expressão é uma lista contínua e fin

Sugestão: Experimente com as expressões 111, 1111, 111111.



3.4 Computabilidade de Turing

Definição

Uma função é Turing-computável se existir um máquina de Turing que a compute. A classe de todas as funções Turing-computáveis é designada por $\mathbf{T\ C}$.

Teorema: $\mathbf{R = C = T\ C}$

As classes das funções parciais recursivas, computáveis por URM e Turing-computáveis são equivalentes.

3.7 Tese de Church

Relembremos a noção intuitiva e informal da classe das

funções efectivamente computáveis: classe de funções cujos valores podem ser calculador por algoritmos ou procedimentos efectivos.

Tese de Church (ou Church-Turing)

A classe intuitiva e informal das funções efectivamente computáveis coincide exactamente com a classe $\mathbf{C}$ das funções computáveis por URM.

3.7 Tese de Church

A Tese de Church não é um teorema susceptível de demonstração ma
Tem um estatuto de crença, cuja veracidade é baseada em evidências:

- Diferentes propostas independentes para uma caracterização formal de computabilidade conduziu à mesma classe de funções, designada por **C**.
- Mostrou-se que uma vasta coleção de funções efectivamente computáveis pertencia a **C**.
- A implementação de um programa URM é um exemplo de um algoritmo. Então, pela própria definição de **C,** todas as funções desta classe são computáveis no sentido informal.
- Nunca ninguém encontrou uma função que fosse computável no sentido informal e não pertencesse a **C.**

3.7 Tese de Church

A Tese de Church é aceite pela grande maioria dos cientistas das mais diversas áreas, mas é mais um resultado científico do que um resultado matemático.

Um dia, pode ser negado por uma nova evidência.

Aplicação da Tese de Church

À luz desta evidência é comum usar a Tese de Church para provar que uma função é computável por URM, através de um algoritmo que a calcule. Este algoritmo pode ser explicitado através de uma linguagem relativamente informal, mas muito rigorosa.

A este procedimento chama-se ***prova pela Tese de Church***.

3.7 Tese de Church

É fundamental ter em consideração o seguinte:

- É ainda necessário provar com rigor que o algoritmo:
  - computa a função correcta;
  - pára em tempo finito, quando a função está definida;
  - pode ser representado por um programa de comprimento finito;

- Quando se usa uma prova pela Tese de Church deve-se estar preparado para construir (ou provar a existência de) um programa URM, uma máquina de Turing ou uma função parcial recursiva equivalente ao algoritmo proposto.

O paradoxo de Russel

Um juiz condenou um homem por um crime reprovável e pretende que o castigo seja particularmente severo. O homem é condenado à morte e a sen processada da seguinte forma:

"A sentença será executada até ao próximo Sábado e, na manhã do dia da execução não saberá de que o dia da morte chegou. Assim, quando o vierem buscar, será uma surpresa".

O prisioneiro, quando ouviu a sentença, respondeu: "*Bem, estou aliviado, pois de acordo com a V. sentença, eu não posso ser executado no Sábado*".

"*Porquê?*" perguntou o juiz. "*Porque, se a sentença deve ser executada até Sábado, e se de facto chegarmos a Sábado, eu saberei que é o dia da minha morte e não será uma surpresa.*"

"*Tem razão*", disse o juiz, "*não poderá ser executado no Sábado. No entanto, não percebo porque está aliviado*".

O paradoxo de Russel (cont.)

"*Bem*", disse o prisioneiro, "*porque se definitivamente não posso ser executado no Sábado, pela mesma razão também não posso ser executado na sexta-feira. Mais ainda, se continuarmos o raciocínio, devo concluir que não posso sequer ser executado hoje*".

O juiz coçou a cabeça e enviou o prisioneiro de volta para a cela. Na quinta-feira, foi executado. Ficou muito surpreendido.

Logo, as ordens do juiz foram cumpridas com sucesso.

Seja **A** um conjunto com a seguinte propriedade: **A** contém todos os conjuntos que não são membros de si próprios.

Pergunta: **A** é membro de si próprio?

O paradoxo de Russel (cont.)

Como Russel resolveu todos os paradoxos:

Russel redefiniu o conceito de ***conjunto*** :

No sentido estrito

Um conjunto tem uma definição que permite a construção de um programa que determina se uma dada entidade é membro do conjunto em tempo finito.

De acordo com esta definição A não é um conjunto e o paradoxo é eliminado.

No sentido lato

O programa que define as regras de pertença a um conjunto não necessita de terminar em tempo finito. Apenas necessita de dar uma resposta em tempo finito. Pode alterar a sua resposta à medida que o programa corre.

Assim, A será membro de si próprio num dado instante e não o ser
resposta nunca é Sim e Não ao mesmo tempo. Alterna entre as duas.

Implicações da Tese de Church

Mas, resolvidos os paradoxos, existem ainda problemas insolúveis.

Turing provou a existência de problemas bem definidos, com uma ú
(que se pode provar que existe), mas que não pode ser calculada por uma Máquina de Turing.

Church e Turing procuram demonstrar que esta simples máquina pode resolver qualquer problema que uma máquina é capaz de resolver. Se um pro
pode ser resolvido por uma máquina de Turing, também não pode ser resolvido por nenhuma outra máquina (… nem pelo homem).

Mais ainda,

de acordo com Church e Turing, cabe-nos a nós, humanos, definir um problema, provar que tem uma única solução, encontrá-la, e ainda provar que nenhuma máquina de Turing é capaz de o resolver.

Ainda ninguém conseguiu...

Implicações da Tese de Church

Os dois lados da Tese de Church:

Negativo

Os problemas que não podem ser resolvidos por uma máquina de Turing também não podem ser resolvidos pelo raciocínio humano. Aceitando esta tese, existem problemas cujas soluções existem e não podem nunca ser encontradas.

Positivo

Se o homem pode resolver um dado problema, então é possível construir uma máquina que o resolva também.

A inteligência artificial é uma tentativa de fornecer uma aplica
de Church.