EINF-2921 Inteligência Artificial **Primavera 2024**

Aula 1: 13.02.24

Professor: Manuel Pita *TA: Zuil Pirola*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

Máquinas de Turing

As Máquinas de Turing, concebidas por Alan Turing em 1936, são um marco fundamental na história da ciência da computação e da inteligência artificial (IA). Este modelo teórico de computação não só desempenhou um papel crucial na compreensão dos limites do que pode ser computado, mas também na definição do próprio conceito de algoritmo, estabelecendo as bases para a era da computação moderna.

A Máquina de Turing é também uma das duas origens da Inteligência Artificial (como veremos em maior detalhe durante as aulas). O formalismo de Turing propõe que é possível 'mecanizar' a inteligência através da implementação do modelo proposto por ele. A abordagem por trás do formalismo assume que podemos definir, e mecanizar, a execução de algoritmos em termos de 'código' que manipula 'símbolos' através de regras específicas, de forma similar a como funciona um programa feito em Python. O programa recebe um input e produz um output que nós conseguimos interpretar, que é útil, mas o computador não sabe que conceitos estão por trás do código. Claramente o cérebro humano não funciona dessa forma. As nossas capacidades cognitivas, a nossa capacidade de abstração e conceptualização do mundo que nos rodeia tem como base as milhares de interligações entre milhares de neurónios. A outra origem da inteligência artificial surge de um formalismo que tenta capturar a forma como as redes neuronais funcionam, tal como veremos nas aulas.

Voltando ao nosso foco neste caderno, a Máquina de Turing, e para pareceber melhor a origem de intelências artificiais baseadas na manipulação de símbolos, temos de entender como funciona esta máquina.

Elementos de uma Máquina de Turing

Uma Máquina de Turing tem componentes que podemos classificar como o *hardware*:

- Uma fita que é dividida em células (como um rolo de papel higiénico). Cada célula (quadrado de papel) pode conter um símbolo, que pode ser lido e escrito pela máquina, incluindo o símbolo vazio.
- Um cabeçalho de leitura/escrita que lê os símbolos na fita e pode mover-se para a esquerda ou para a direita, uma célula de cada vez.

e outras componentes que podemos classificar como *software*:

- Um conjunto de símbolos (definidos pelo programador da máquina) que permitem escrever entradas e saídas na fita. Por exemplo, se o programador quiser implementar uma calculadora simples, precisará dos números 0-9 e os símbolos para somar, restar, etc.

- Um conjunto de estados internos, incluindo um estado inicial e estados finais (ou de aceitação). Estes estados são um dos aspectos mais importantes da genialidade do Turing na implementação da sua máquina. Os estados são de alguma forma parecidos aos símbolos, mas estes não são para escrever entradas e saídas na fita. Os estados determinam partes do procedimento que esta a ser implementando. Por exemplo, quando fazemos uma soma de números manualmente, coluna a coluna, de direita a esquerda, e o nosso resultado parcial é maior que dez, internamente guardamos uma nota "vai um" que utilizamos a seguir no resultado da seguinte coluna. É exatamente esta função auxiliar a que é cumprida pelos estados. Quanto mais complexo o procedimento, mais estados internos serão necessários. Os estados internos são guardadados no cabeçalho da máquina.
- Uma tabela de transição que define, para cada combinação possível de estado interno e símbolo na fita (vamos referir a estes pares pelo termo *condições*), três coisas (que determinam em cojunto a *ação* a executar quando a máquina se encontra numa determinada condição): o (a) símbolo a ser escrito; (b) a direção para mover o cabeçalho, e (c) o próximo estado interno da máquina.

Exemplo: Consideremos uma Máquina de Turing simples que consegue determinar se un *input* representado como uma cadeia de caracteres binários contêm, ou não, um número par de zeros. O programador da máquina determina que a mesma irá precisar de três estados como mínimo para funcionar: $Q = \{q_0, q_1, q_f\}$, onde q_0 é o estado inicial, q_1 é um estado intermediário utilizado para que a máquina possa memorizar o ter "visto" um zero, e q_f é o estado final ou de paragem, o qual determina que o que estiver escrito na fita é o *output* e faz com que a máquina pare. A tabela de transição pode ser definida da seguinte forma para os símbolos 0 e 1, assumindo que B representa um espaço em branco (ver também a Figura 1.1):

- Se no estado q_0 lê-se um 0, escreve-se um B, move-se para a direita, e transita-se para o estado q_1.
- Se no estado q_1 lê-se um 0, escreve-se um B, move-se para a direita, e transita-se para o estado q_0.
- Se no estado q_0 ou q_1 lê-se um 1, escreve-se um B, move-se para a direita, e permanece-se no mesmo estado em que estava.
- Se no estado q_0 lê-se um B (indicando o fim do input), escreve-se 0, move-se a esquerda e transita-se para o estado q_f.
- Se no estado q_1 lê-se um B (indicando o fim do input), escreve-se 1, move-se a esquerda e transita-se para o estado q_f.
- no estado q_f a máquina para. Se o output é zero, então o número de zeros é par, caso contrário se o output é um significa que o número de zeros no input é impar. É importante notar que estas interpretações são feitas pelo programador e por quem usar a máquina: a máquina não sabe nada sobre números pares ou impares!

Como executar uma Máquina de Turing?

Para executar a Máquina de Turing do exemplo anterior, siga os passos abaixo:

1. Inicie no estado q_0 com o cabeçalho de leitura/escrita posicionada no primeiro símbolo da cadeia de entrada (o símbolo mais a esquerda).
2. Leia o símbolo sob o cabeçalho. Utilize a tabela de transição para determinar a ação a ser tomada (escrever um símbolo, mover a cabeça, e mudar de estado).

3. Repita o passo 2 até que a máquina atinja o estado final q_f ou até que não haja mais transições possíveis, indicando que a cadeia de entrada não é aceite pela máquina.

Este processo demonstra como a Máquina de Turing pode decidir sobre a aceitação de cadeias de caracteres seguindo um conjunto definido de regras. Qualquer cadeia de entrada que faz com que a máquina atinja o estado q_f é uma entrada válida para qual o output poderá ser computado.

O Contexto Filosófico e Prático

Desde o início, as Máquinas de Turing foram envoltas em questões tanto filosóficas quanto práticas. Filosoficamente, elas abordam o problema da "decidibilidade" - isto é, quais problemas podem ser resolvidos por meio de procedimentos computacionais.

A decidibilidade é um conceito fundamental na teoria da computação que se refere à capacidade de um sistema computacional, como uma Máquina de Turing, resolver um problema de maneira definitiva. Em termos simples, um problema é considerado "decidível" se, para qualquer entrada dada, a máquina consegue sempre chegar a uma resposta de "sim" ou "não" após um número finito de passos. Este conceito é crucial para entender os limites do que pode ser computado: nem todos os problemas são decidíveis, o que significa que existem questões para as quais não é possível construir um algoritmo que sempre chegue a uma solução.

Turing demonstrou que tal algoritmo não pode existir, provando assim a existência de problemas indecidíveis. A importância deste resultado não pode ser subestimada: ele estabelece limites fundamentais para a computação, mostrando que existem questões que estão além da capacidade de qualquer máquina de Turing, e por extensão, além do que podemos computar.

A introdução do "Problema da Parada" por Turing, que demonstra a existência de questões que não podem ser resolvidas por algoritmos, desafia nossa compreensão dos limites da computação e, por extensão, da inteligência humana e artificial.

O Problema da Parada–*halting problem*–introduzido por Alan Turing, é um exemplo clássico de um problema indecidível. Ele questiona se é possível criar um algoritmo que, ao receber como entrada qualquer programa de computador e uma entrada inicial para esse programa, possa determinar corretamente se o programa termina (ou "para") ou continua a executar indefinidamente.

A decidibilidade e o Problema da Parada têm implicações profundas não só na teoria da computação, mas também na filosofia da mente, na lógica e em outras áreas da ciência e da matemática. Eles nos obrigam a reconhecer que, apesar do imenso poder dos computadores e dos algoritmos, existem limites claros para o que é computacionalmente possível. Este reconhecimento é fundamental para o desenvolvimento de teorias computacionais, a prática da programação e o avanço da inteligência artificial, uma vez que nos ajuda a identificar quais problemas são acessíveis à solução algorítmica e quais permanecerão fora do alcance.

Pragmaticamente, a abstração da Máquina de Turing permitiu o desenvolvimento de computadores digitais e linguagens de programação. Ao fornecer uma estrutura para entender o processamento de dados e a execução de instruções, as Máquinas de Turing influenciaram diretamente a criação de sistemas computacionais reais e a evolução do software.

Impacto na Inteligência Artificial

No campo da inteligência artificial, as Máquinas de Turing têm uma relevância especial. Elas fornecem um modelo teórico para entender a computação algorítmica, que está no coração da IA. A ideia de que uma

máquina pode simular qualquer processo de pensamento computacional é fundamental para o desenvolvimento de algoritmos de IA, desde simples automações até complexos sistemas de aprendizado de máquina.

Por Que Entender as Máquinas de Turing Importa

Entender as Máquinas de Turing é crucial por várias razões:

- **Fundamentos Teóricos:** Elas oferecem uma compreensão básica dos princípios da computação, essencial para qualquer pessoa que trabalhe com tecnologia da informação, ciência da computação e IA.
- **Limites da Computação:** Compreender o que as Máquinas de Turing podem e não podem fazer ajuda a definir o escopo do possível em tecnologia e IA, guiando pesquisadores e praticantes nas suas inovações.
- **Evolução da IA:** O modelo de Turing é um lembrete constante da origem algorítmica da inteligência artificial, inspirando o desenvolvimento de novas abordagens e técnicas na área.

A história das Máquinas de Turing é, portanto, não apenas uma crônica da ciência da computação, mas também um capítulo vital na contínua busca pelo entendimento da inteligência, seja ela humana ou artificial. Através desta lente, estudantes e profissionais podem apreciar melhor a beleza e os desafios da computação e da IA, equipados com um conhecimento profundo que transcende a tecnologia para tocar em questões fundamentais de lógica, filosofia e capacidade humana.

Sessão prática

Durante a sessão prática devem implementar código Python para definir e executar uma máquina de Turing. Para tal terão de definir uma forma de representar o programa (condições e ações) utilizando, por exemplo, un ficheiro `.CSV` e guardar numa variável adequada `program`. A seguir terão de escrever uma função `get_action(condition, program)` que dada uma condição devolve a ação que deve ser executada. No passo seguinte terão de pensar em como representar a máquina, nomeadamente a fita com o seu input, onde está o cabeçalho e o estado interno inicial. Logo vão precisar duma função `update_machine` que, dado o programa e a máquina no seu estado atual, devolva a máquina depois de executar uma ação. A última função que vão precisar será `run_machine` a qual recebe a máquina no seu estado inicial e o programa, e executa cada paso, imprimindo todas as informações (fita, posição do cabeçalho e estaod interno) até atingir o estado final, ou um estado para o qual não há ação (breakpoint). Há várias formas válidas de implementar uma máquina de Turing.

A

$$S = \{0, 1, B\}$$

$$Q = \{q_0, q_1, q_f\}$$

B

Current symbol	Current state	Write	Move	Next state
0	q_0	B	R	q_1
0	q_1	B	R	q_0
0	q_f			
1	q_0	B	R	q_0
1	q_1	B	R	q_1
1	q_f			
B	q_0	0	L	q_f
B	q_1	1	L	q_f
B	q_f		Halt	

C

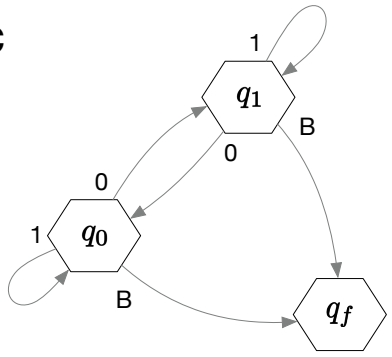

Figure 1.1: **Máquina de Turing para determinar de input binário tem número par de zeros.** (A) esta Máquina de Turing considera unicamente três símbolos para representar inputs e outputs na fita $S = \{0, 1, B\}$, e considera três estados internos: q_0 representa máquina em reset, o qual acontece no início e cada vez que a máquina encontrou, até onde estiver o cabecalho, um número par de zeros; q_1 representa que a máquina encontrou, até onde estiver o cabecalho, um número impar de zeros; finalmente q_f representa que a cadeia de entrada foi processada completamente. (B) A partir dos símbolos e dos estados internos, temos de producir uma tabela de instruções que inicialmente contêm todos os pares possíveis de símbolo e estado. Cada um destes pares é referido como *condição*. Para cada condição útil na execução do algoritmo, o programador define uma acção que deve ter três partes: o que o cabecalho tem de escrever na fita, para onde se vai mover (esquerda ou direita), e qual o seu novo estado interno. É importante destacar que esta máquina específica não precisou usar todas as condições. Note que, por exemplo, a máquina não considera possível o encontrar-se numa situação na qual esteja a ler o símbolo 0 no estado q_f. Um programa que funciona numa máquina de Turing não precisa de usar todas as condições possíveis definidas pelos conjuntos de símbolos e estados internos. (C) uma forma adequada de representar e avaliar máquinas no geral é conhecida como *diagrama de transição de estados* que vamos estudar em detalhe nas nossas aulas.

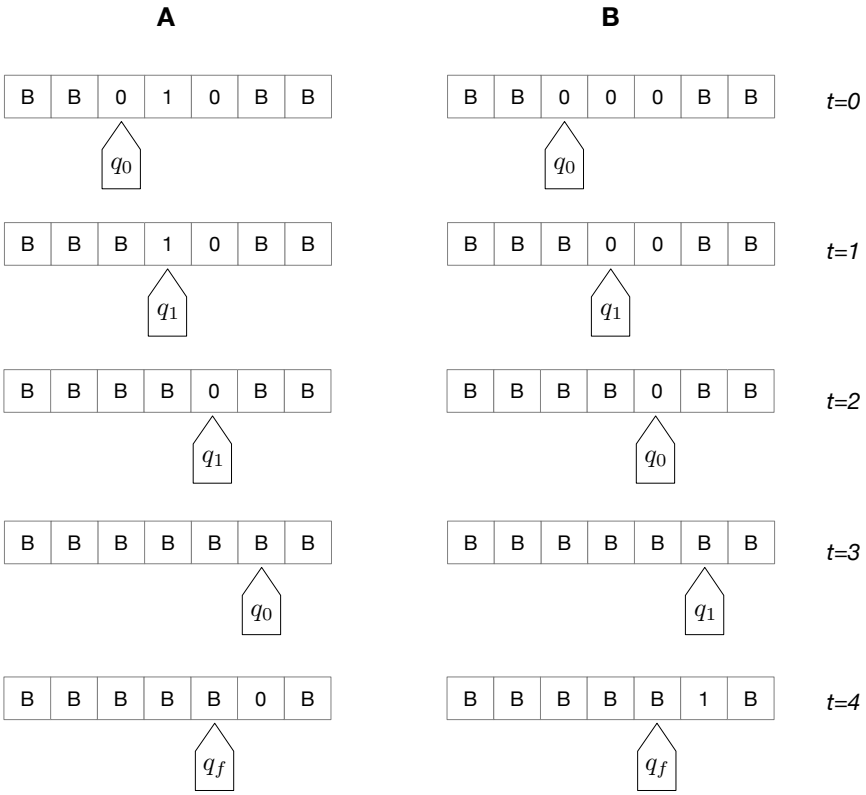

Figure 1.2: **Execução da Máquina de Turing para dois inputs diferentes.** (A) Execução da máquina para o input 010, a qual depois de quatro passos atinge o estado final, com output 0 (número par de zeros). (B) Execução da máquina para o input 000 a qual depois de quatro passos atinge o estado final, com output 1 (número impar de zeros). Quando a máquina chega ao estado q_f executa uma instrução especial: *Halt* que faz com que a máquina pare.