Teoria da Computação

Aula Teórica 22

Máquinas de Turing

António Ravara

Departamento de Informática

27 de Maio de 2019

Máquina de Turing

- Definida em 1936 por Alan Turing
  licenciou-se em Matemática em Cambridge em 1934;
  tinha 23 anos quando escreveu o artigo!.
- O primeiro "aparelho computacional automático":
  captura a "essência" da computação.
- Poder: calcula automaticamente
  QUALQUER função computável.
- Todo o aparelho computacional é simulado por uma
  Máquina de Turing:
  - NADA é mais "potente" computacionalmente!
    É a "Church-Turing thesis"
    Church em Princeton era o lógico mais importante da área; tinha uma definição matemática equivalente (mas não "efectivamente computável")
  - Os nossos smartphones, tablets, computadores, etc,
    "são" Máquinas de Turing *universais*.

Máquina de Turing: modelo *preciso e conciso* do que é, e do que pode fazer,um computador

Uma idealização matemática - um modelo

Uma Máquina de Turing é um programa.
Há Máquinas de Turing que "processam" outras Máquinas de Turing.
O que é um compilador?

ACE: Automatic Computing Engine

- Desenhado por Turing em 1946.
- É o primeiro computador electrónico programável.
- Foi considerado muito ambicioso e caro – £11200
  (o Reino Unido estava em crise económica)
- Devido aos segredos de guerra (Official Secrets Act)Turing não pode explicar porque sabia que era realizável.
  Só foi construido (numa versão simplificada) em Manchester em 1950.

Função computável

- Uma máquina de Turing implementa uma função:
  dado um argumento (arbitrário), calcula o resultado.
- A máquina é um *algoritmo*:
  Um procedimento finito para calcular o resultado a partir de qualquer *input* válido.
- Tese de Church-Turing:
  - qualquer função tem um algoritmo se, e só se, é computável.
  - Funções computáveis são as mecanicamente calculáveis, com tempo e memória ilimitados.
- Já vimos que há funções não computáveis:
  Exemplo: o predicado que captura o paradoxo do mentiroso ("só digo mentiras").
- Aparentemente, há máquinas de Turing na natureza:
  - O Ribossoma ("tradutor"de RNA em proteínas) "é" uma Máquina de Turing!
  - Mas o seu mecanismo só foi descrito na segunda metade do séc XX...

Universalidade

- O modelo das máquinas de Turing diz-se *universal* porque qualquer função computável tem uma máquina que a calcula (é o seu *algoritmo*) (e vice-versa).
- Consequência: *incompletude* (ou indecidibilidade) computacional: Há funções não computáveis.
- Diz-se *Turing-completo*, qualquer mecanismo que permite implementar qualquer função computável.
- A maioria dos processadores *genéricos* (como o 8080 ou o i7) são Turing-completos.
- As linguagens de programação *genéricas* (como o Java ou o C) são Turing-completas. O SQL não é...

Máquina de Turing Universal

Simula uma máquina de Turing arbitrária com argumento arbitrário.

É a base da *arquitectura de von Neumann* (1946), que está em TODOS os computadores!

Stack-Based Turing Machine: dados

O modelo apresentado não é o original, mas permite programas mais compactos e semelhantes aos de uma máquina abstracta como a do Java.

- O dados da máquina são *árvores binárias*.
- As folhas são símbolos de um alfabeto Σ.
- Assume-se que o alfabeto contém o símbolo `null`, que denota a árvore vazia.
- O conjunto *DATA* é definido indutivamente como se segue:
  - $a \in \Sigma \longrightarrow a \in DATA$
  - $s \in DATA \wedge t \in DATA \longrightarrow (s, t) \in DATA$

  O par (s, t) representa a árvore que tem s como subárvore esquerda e t como subárvore direita.
- Como já se viu (no caso dos pares), podem-se representar listas como árvores:
  $[1, 2, 3]$ é representada como $(1, (2, (3, \texttt{null})))$.

Stack-Based Turing Machine: componentes

- Uma pilha, de elementos de *DATA*.
  Assume-se que quando a máquina começa a operar, a pilha está vazia.
- Uma sequência, potencialmente infinita, de células de memória.
  Cada uma pode conter um elemento de *DATA*.
  Assume-se que quando a máquina começa a operar, todas as células contêm o valor `null`
  (chama-se a tal configuração de memória $M_{\texttt{null}}$).
- Um contador de programa, que contém o estado em que a máquina está em cada momento da computação.
  Assume-se que quando a máquina começa a operar, o contador contém o estado inicial.
- Uma função de transição, $\tau \subseteq S \times OP \rightarrow S$, sendo:
  - S um conjunto finito de estados, sendo um o inicial e $F \subseteq S$ o conjunto dos estados finais;
  - OP um conjunto de operações.

Stack-Based Turing Machine: operações

- `push t` : coloca t no topo da pilha.
- `left` / `right` : se a árvore no topo da pilha não for nem vazia nem singular, substitui-a pela sua sub-árvore esquerda / direita.
- `cons` : retira dois valores do topo da pilha, sendo t_1 o primeiro e t_2 o segundo, e coloca no topo da pilha a árvore (t_1, t_2).
- `eq` : retira dois valores do topo da pilha e compara-os; se forem iguais coloca `true` no topo da pilha; se forem diferentes coloca `false` no topo da pilha.
- $?a$: executa a transição se o valor no topo da pilha é a; caso contrário não executa a transição.
- ?`cons` : executa a transição se o valor no topo da pilha é uma árvore não vazia; caso contrário não executa a transição.
- `load` k : coloca no topo da pilha o valor na célula de memória M_k.
- `store` k : retira o valor no topo da pilha e coloca-o na célula de memóra M_k.

Stack-Based Turing Machine: configurações e transições

- Uma configuração da máquina é um triplo (M, p, q), onde:
  - M é a memória; p é a pilha; e q é o estado de controle.
  - Uma pilha p com n elementos (sendo $n \in \mathbb{N}_0$) escreve-se como $\langle s_1, \ldots, s_n \rangle$ onde $\forall_{0 \leq i \leq n} s_i \in \mathit{DATA}$.
  - $M[k]$ denota o conteúdo da célula M_k.
  - $M[k/v]$ representa a memória que contém os valores da M em todas as células, excepto na k, que contém v.
- Sejam $q, q' \in S$, $a, b \in \Sigma$, e $s, t, l, r, u \in \mathit{DATA}$.
  A função de transição em configurações é dada por:

$$
\begin{array}{ll}
(M, s, q) \longrightarrow (M, (t, s), q') & \text{se } (q, \texttt{push } t, q') \in \tau \\
(M, ((l, r), s), q) \longrightarrow (M, (l, s), q') & \text{se } (q, \texttt{left}, q') \in \tau \\
(M, ((l, r), s), q) \longrightarrow (M, (r, s), q') & \text{se } (q, \texttt{right}, q') \in \tau \\
(M, (l, (r, s)), q) \longrightarrow (M, ((l, r), s), q') & \text{se } (q, \texttt{cons}, q') \in \tau \\
(M, (a, s), q) \longrightarrow (M, (a, s), q') & \text{se } (q, \texttt{?}a, q') \in \tau \\
(M, ((a, b), s), q) \longrightarrow (M, ((a, b), s), q') & \text{se } (q, \texttt{?cons}, q') \in \tau \\
(M, s, q) \longrightarrow (M, (M[k], s), q') & \text{se } (q, \texttt{load } k, q') \in \tau \\
(M, (u, s), q) \longrightarrow (M[k/u], s, q') & \text{se } (q, \texttt{store } k, q') \in \tau
\end{array}
$$

Stack-Based Turing Machine: computações

- Uma computação da SBTM é uma sequência finita de transições:

$$(M_{initial}, \langle\rangle, q) \longrightarrow^* (M_{final}, p, q')$$

onde

- q é o estado inicial e $q' \in F$ é um estado de controle final;
- $M_{initial}$ é o estado inicial da memória, M_{final} é o estado final da memória.
- na configuração inicial, a pilha está vazia;
  no fim pode conter informação (depende da intenção do programador).

Stack-Based Turing Machine: a definição

Uma Stack-Based Turing Machine T é a estrutura

$T = \langle S, \Sigma, s, M, \tau, F \rangle$

onde:

- S é o conjunto (finito) dos estados de controle;
- Σ é o alfabeto e $DATA$ o conjunto dos valores manipulados;
- $s \in S$ é o estado inicial;
- M é a memória inicial;
- $\tau \subseteq S \times OP \times S$ é a relação de transição.
  Consideramos apenas relações deterministas:
  $$((s, op_1, s') \in \tau \wedge (s, op_2, s'') \in \tau \wedge s' \neq s'') \longrightarrow (op_1 = ?\alpha \wedge op_2 = ?\beta \wedge \alpha \neq \beta)$$
  $$OP = \{\texttt{push}\ t, \texttt{left}, \texttt{right}, \texttt{cons}, \texttt{eq}, ?t, ?\texttt{cons}, \texttt{load}\ k, \texttt{store}\ k\}$$
  com $t \in \texttt{DATA}$ e $k \in NAT$.
- $F \subseteq S$ é o conjunto de estados finais.