Computabilidade

Linguagens de Programação 2016.2017

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

Funções totais e parciais

Funções não computáveis

Halting problem

Programa

É uma função

```
output = function( input, estado_maq )
```

Resultado

- Expressão com valor
  - 3+2
- Expressão sem valor
  - Terminação por erro
    - 3/0
  - Não terminação
    - f(5), com
    - f(x)=if (x=0) then 0 else x+f(x-2)

Funções totais e parciais

Função total

$f : A \to B$ é um conj. $f \subseteq A \times B$ com

Se $\langle x,y \rangle \in f$ e $\langle x,z \rangle \in f$, então y=z

Para cada $x \in A$, existe um $y \in B$ com $\langle x,y \rangle \in f$

valor único

total

Função parcial

$f : A \to B$ é um conj. $f \subseteq A \times B$ com

Se $\langle x,y \rangle \in f$ e $\langle x,z \rangle \in f$, então y=z

valor único

Programa == função parcial

Operações parciais

Não terminação

Computabilidade

Função computável

Uma função `f` é **computável** se existir um programa `P` que calcula `f`

Para qualquer entrada `x`, o cálculo `P(x)` termina com saída `f(x)`

Função parcial recursiva

É uma função parcial computável (de inteiros para inteiros)

Função não computável

Função não computável

função para a qual não existe um programa `P` que a calcule

Exemplo

Halting problem

Decidir se um programa termina com input `x`

Halting problem

Função

Dado um programa `P`, uma entrada `x`

Determinar se `P` termina com `x`

$$Halt(P,x) = \begin{cases} \text{"termina"} & \text{se } P(x) \text{ termina} \\ \text{"não termina"} & \text{se } P(x) \text{ não termina} \end{cases}$$

Esta função ***`Halt()`*** é não computável!

Não existe nenhum programa que calcule *`Halt()`*

Demonstração (1)

1. Assumir a existência de tal programa

```
Q → string
```

$$Q(P,x) = \begin{cases} \text{"termina"} & \text{se } P(x) \text{ termina} \\ \text{"não termina"} & \text{se } P(x) \text{ corre para sempre} \end{cases}$$

2. Construir um programa *D a partir de Q*

$$D(P) = if\ Q(P,P) = \text{"termina"}\ then\ \textbf{corre para sempre}\ else\ \textbf{termina}$$

O programa *D* tem o seguinte comportamento

$$D(P) = \begin{cases} \text{termina} & \text{se } P(P) \text{ corre para sempre} \\ \text{corre para sempre} & \text{se } P(P) \text{ termina} \end{cases}$$

Demonstração (2)

O programa *D* tem o seguinte comportamento

$$D(P) = \begin{cases} \text{termina} & \text{se } P(P) \text{ corre para sempre} \\ \text{corre para sempre} & \text{se } P(P) \text{ termina} \end{cases}$$

3. O que faz *D(D)*?

Se $D(D)$ termina, então é porque $D(D)$ corre para sempre
Se $D(D)$ corre para sempre, então é porque $D(D)$ termina

Contradição!!!

4. A suposição inicial é falsa!

assumir que existe um programa Q que resolve o "*halting problem*"

Implicações

Existem propriedades úteis dos programas que não se conseguem determinar

- O programa vai correr para sempre?
- O programa vai (eventualmente) causar um erro?
- O programa vai aceder novamente a um espaço específico de memória?

Exemplo

```
i=0;
while (i != f(i))
    i = g(i);
printf(...i...);
```