

# Semântica das Linguagens de Programação

Maria João Frade  
[mfj@di.uminho.pt](mailto:mjf@di.uminho.pt)

1

# Apresentação

2

## Semântica das Linguagens de Programação

- Introdução à **semântica formal** dos programas
- Dar uma **descrição precisa do significado** dos programas
  - revela ambiguidades e subtilezas
  - lança as bases para a implementação, análise e verificação de programas
- Apresentação de **várias abordagens** à semântica
  - as ideias fundamentais (c/ uma linguagem de programação simples)
  - o seu inter-relacionamento
  - a sua aplicabilidade

3

## Temas

- **Semântica de Programas Imperativos**
  - Semântica operacional (natural e estrutural)
  - Semântica denotacional
  - Semântica axiomática
  - Relações entre semânticas
  - Máquinas abstractas
- **Lambda-calculus** (puro e com tipos)
- **Semântica de Programas Funcionais**
  - Sistema de tipos
  - Semântica de avaliação “call-by-value”
  - Semântica de avaliação “call-by-name”

4

# Bibliografia

- **Semantics with Applications: A formal introduction.**  
H. Nielson & F. Nielson. Wiley, 1992. [[disponível online](#)]  
(re-editado pela Springer em 2007)
- **Theories of Programming Languages.**  
J.C. Reynolds. Cambridge Univ. Press, 1998.
- **The Semantics of Programming Languages.**  
M. Hennessy. Wiley, 1990. [[disponível online](#)]
- **The Formal Semantics of Programming Languages.**  
G. Winskel. MIT Press, 1993.

5

# Avaliação

- Dois testes, 50% cada (nota mínima: 5 valores)
  - 1º teste: ???
  - 2º teste: ???
- Exame de recurso: ???

6

# Def. de uma Linguagem de Programação

- **Sintaxe:** focada na estrutura gramatical do programa. Descrição dos símbolos e das expressões que constituem a linguagem, bem como das demais categorias sintácticas (comandos, programas, ...)
- **Semântica:** descrição do significado das várias construções da linguagem, que permite prever o seu comportamento em tempo de execução. Lida apenas com programas sintaticamente correctos.
- **Ferramentas:** parsers, interpretadores, compiladores, debuggers, profilers...

7

# Semântica Informal

Qual o significado da seguinte construção?

**while** *B* **do** *C*

Segundo Kernighan & Ritchie, em *The C Programming Language*,

*“O comando C é executado repetidamente enquanto o valor da expressão B for verdadeiro, sendo o teste feito antes da execução do comando.”*

8

# Semântica Formal

- É a construção de um modelo matemático.
- Preocupa-se em especificar rigorosamente o comportamento dos programas.
- Especificação do comportamento é independente da máquina.
- Lança as bases para a implementação, análise e verificação dos programas.
- Permite revelar vários tipos de subtilizas de que é importante estar ciente.

9

# Benefícios da Semântica Formal

- **No desenho da linguagem:** pode ser útil na clarificação de aspectos subtils e na descoberta de melhores formas de organizar a informação.
- **Na implementação:** permite processar, analisar e optimizar os programas de forma correcta. Lança as bases para a definição de máquinas abstractas e código intermédio.
- **Na verificação:** permite raciocinar acerca dos programas, especificações e outras propriedades.

10

# Abordagens à Semântica Formal

- **Operacional**
  - Descreve os passos de execução de um programa como uma máquina abstracta.
  - Foca-se em *como* o efeito de uma computação é produzido.
- **Denotacional**
  - Descreve o significado de um programa num domínio matemático abstracto.
  - Foca-se apenas *no* efeito da computação não em como ele é obtido.
- **Axiomática**
  - Descreve o comportamento de programas através de uma lógica.
  - Foca-se nas *propriedades* do efeito de executar um programa.

11

# Cada estilo tem os seus usos...

- **Operacional:** útil na implementação das linguagens.
- **Axiomática:** útil na verificação dedutiva dos programas.
- **Denotacional:** útil para trabalhar a um nível mais abstracto.

**Na realidade os vários estilos usam-se uns aos outros.**

Por exemplo:

- A prova de correção de um sistema de inferência axiomático é feita face à semântica operacional ou denotacional.

12

# Indução

13

## Indução

- A indução é uma técnica que nos permite definir e raciocinar com objectos que são
  - *estruturados* de uma forma bem fundada,
  - *finitos* embora arbitrariamente grandes.
- A indução explora a natureza finita e estruturada desses objectos para superar a sua complexidade arbitrária.
- Os objectos compostos tem uma forma única de ser decompostos nos seus constituintes imediatos.

Na abordagem formal à semântica as definições indutivas e as provas por indução estão por todo o lado...

14

## Indução Matemática

Para qualquer propriedade  $\Phi(x)$  sobre números naturais  $x \in \mathbb{N} \stackrel{\text{def}}{=} \{0, 1, 2, \dots\}$

para provar  $\forall x \in \mathbb{N}. \Phi(x)$

basta provar

$$\begin{array}{ll} \Phi(0) & \text{(caso base)} \\ \text{e } \forall x \in \mathbb{N}. \Phi(x) \Rightarrow \Phi(x + 1) & \text{(passo indutivo)} \end{array}$$

15

## Boa fundação

Uma relação binária  $\prec$  sobre um conjunto  $A$  diz-se uma **relação bem fundada** se não existir nenhuma sequência infinita decrescente  $\dots \prec a_3 \prec a_2 \prec a_1$

A seguinte relação definida sobre um conjunto definido indutivamente é bem fundada

$$t' \prec t \text{ sse } t' \text{ é um sub-termo estrito de } t$$

### Indução bem fundada

Seja  $\Phi$  uma propriedade sobre elementos de um conjunto  $A$  definido indutivamente.

Para provar  $\forall a \in A. \Phi(a)$

basta assumir  $\Phi(a')$  para todo o  $a' \prec a$  e com isso provar que  $\Phi(a)$  se verifica.

16

# Indução Estrutural

Prova de uma propriedade por indução estrutural numa determinada categoria sintática.

## Casos de base

- Provar que a propriedade se verifica para todos os elementos de base da categoria sintática.

## Casos indutivos (para cada elemento composto da categoria sintática)

- Assumir que a propriedade se verifica para todos os constituintes imediatos do elemento (estas são as *hipóteses de indução*) e provar que esta também se verifica para o elemento composto.

17

## Uma prova por indução estrutural

Provar que  $\mathcal{N}: \text{Num} \rightarrow \mathbf{Z}$  é uma função total.

$\mathcal{N}$  é uma função total, se para todo o argumento  $n \in \text{Num}$  existir um único número  $\mathbf{n} \in \mathbf{Z}$  tal que  $\mathcal{N}[n] = \mathbf{n}$

**Prova:** Por indução na estrutura de  $n$ .

### • Casos de base

Provar a propriedade quando  $n$  é 0 ou 1.

### • Casos indutivos

Provar a propriedade quando  $n$  é  $n'0$  ou  $n'1$ .

19

## Um pequeno exemplo

Considere a representação de números no sistema binário dada pela categoria sintática **Num** de acordo com a seguinte sintaxe abstracta

$$n ::= 0 \mid 1 \mid n\ 0 \mid n\ 1$$

usamos  $n \in \text{Num}$  como meta-variável.

Para determinar o número inteiro representado pelo numeral definimos uma função semântica

$$\mathcal{N}: \text{Num} \rightarrow \mathbf{Z}$$

definida indutivamente da seguinte forma

$$\mathcal{N}[0] = 0$$

$$\mathcal{N}[1] = 1$$

$$\mathcal{N}[n\ 0] = 2 \cdot \mathcal{N}[n]$$

$$\mathcal{N}[n\ 1] = 2 \cdot \mathcal{N}[n] + 1$$

18

## A linguagem **While**

Categorias sintáticas e respectivas meta-variáveis.

$n$  will range over numerals, **Num**, (em notação decimal)

$x$  will range over variables, **Var**,

$a$  will range over arithmetic expressions, **Aexp**,

$b$  will range over boolean expressions, **Bexp**, and

$S$  will range over statements, **Stm**. (os comandos da linguagem)

20

# A linguagem **While**

## Sintaxe abstracta

```
a ::= n | x | a1 + a2 | a1 * a2 | a1 - a2
b ::= true | false | a1 = a2 | a1 ≤ a2 | ¬b | b1 ∧ b2
S ::= x := a | skip | S1 ; S2 | if b then S1 else S2
      | while b do S
```

21

## Estado

O estado associa a cada variável o seu valor corrente.

O **estado** será representado por uma função de variáveis para valores (do domínio de interpretação).

$$\text{State} = \text{Var} \rightarrow \mathbf{Z}$$

Seja  $s$  um estado.

$s[x]$  representa o valor da variável  $x$  no estado  $s$ .

$s[y \mapsto v]$  representa a seguinte função:

$$(s[y \mapsto v])x = \begin{cases} v & \text{if } x = y \\ s[x] & \text{if } x \neq y \end{cases}$$

23

## Semântica das expressões

Os numerais em notação decimal têm o significado esperado.

Por exemplo,

$$\mathcal{N}[137] = 137 \in \mathbf{Z}$$

**Note que:** um numeral é uma entidade sintática e um número é um valor semântico.

O significado de uma expressão depende do valor atribuído às variáveis que nela ocorrem. Por exemplo, a expressão  $x + 1$

- terá o valor 4 se o valor de  $x$  for 3
- terá o valor 3 se o valor de  $x$  for 2

Surge assim a necessidade de introduzir a noção de **estado**.

22

## Semântica das expressões aritméticas

É dada pela seguinte função semântica

$$\mathcal{A}: \mathbf{Aexp} \rightarrow (\text{State} \rightarrow \mathbf{Z})$$

definida indutivamente do seguinte modo

$$\begin{aligned}\mathcal{A}[n]s &= \mathcal{N}[n] \\ \mathcal{A}[x]s &= s[x] \\ \mathcal{A}[a_1 + a_2]s &= \mathcal{A}[a_1]s + \mathcal{A}[a_2]s \\ \mathcal{A}[a_1 * a_2]s &= \mathcal{A}[a_1]s \cdot \mathcal{A}[a_2]s \\ \mathcal{A}[a_1 - a_2]s &= \mathcal{A}[a_1]s - \mathcal{A}[a_2]s\end{aligned}$$

24

## Semântica das expressões booleanas

É dada pela seguinte função semântica

$$\mathcal{B}: \mathbf{Bexp} \rightarrow (\mathbf{State} \rightarrow \mathbf{T})$$

definida indutivamente por

$$\begin{aligned}\mathcal{B}[\text{true}]_s &= \text{tt} \\ \mathcal{B}[\text{false}]_s &= \text{ff} \\ \mathcal{B}[a_1 = a_2]_s &= \begin{cases} \text{tt} & \text{if } \mathcal{A}[a_1]_s = \mathcal{A}[a_2]_s \\ \text{ff} & \text{if } \mathcal{A}[a_1]_s \neq \mathcal{A}[a_2]_s \end{cases} \\ \mathcal{B}[a_1 \leq a_2]_s &= \begin{cases} \text{tt} & \text{if } \mathcal{A}[a_1]_s \leq \mathcal{A}[a_2]_s \\ \text{ff} & \text{if } \mathcal{A}[a_1]_s > \mathcal{A}[a_2]_s \end{cases} \\ \mathcal{B}[\neg b]_s &= \begin{cases} \text{tt} & \text{if } \mathcal{B}[b]_s = \text{ff} \\ \text{ff} & \text{if } \mathcal{B}[b]_s = \text{tt} \end{cases} \\ \mathcal{B}[b_1 \wedge b_2]_s &= \begin{cases} \text{tt} & \text{if } \mathcal{B}[b_1]_s = \text{tt} \text{ and } \mathcal{B}[b_2]_s = \text{tt} \\ \text{ff} & \text{if } \mathcal{B}[b_1]_s = \text{ff} \text{ or } \mathcal{B}[b_2]_s = \text{ff} \end{cases}\end{aligned}$$

25

$\mathbf{T} = \{\text{tt}, \text{ff}\}$  é o conjunto dos valores de verdade

## Variáveis livres

$\text{FV}(a)$  denota o conjunto das *variáveis livres* de uma expressão aritmética  $a$

Define-se indutivamente do seguinte modo

$$\begin{aligned}\text{FV}(n) &= \emptyset \\ \text{FV}(x) &= \{x\} \\ \text{FV}(a_1 + a_2) &= \text{FV}(a_1) \cup \text{FV}(a_2) \\ \text{FV}(a_1 * a_2) &= \text{FV}(a_1) \cup \text{FV}(a_2) \\ \text{FV}(a_1 - a_2) &= \text{FV}(a_1) \cup \text{FV}(a_2)\end{aligned}$$

De modo similar define-se  $\text{FV}(b)$  para expressões booleanas.

26

## Substituições

$a[y \mapsto a_0]$  denota a *substituição*, na expressão  $a$ , de cada ocorrência livre da variável  $y$  pela expressão  $a_0$ .

Define-se indutivamente do seguinte modo

$$\begin{aligned}n[y \mapsto a_0] &= n \\ x[y \mapsto a_0] &= \begin{cases} a_0 & \text{if } x = y \\ x & \text{if } x \neq y \end{cases} \\ (a_1 + a_2)[y \mapsto a_0] &= (a_1[y \mapsto a_0]) + (a_2[y \mapsto a_0]) \\ (a_1 * a_2)[y \mapsto a_0] &= (a_1[y \mapsto a_0]) * (a_2[y \mapsto a_0]) \\ (a_1 - a_2)[y \mapsto a_0] &= (a_1[y \mapsto a_0]) - (a_2[y \mapsto a_0])\end{aligned}$$

27

## Algumas propriedades

Sejam  $s$  e  $s'$  estados tais que  $s[x] = s'[x]$  para todo  $x$  em  $\text{FV}(a)$ . Então,  $\mathcal{A}[a]_s = \mathcal{A}[a]_{s'}$

**Prova:** por indução na estrutura de  $a$ .

Para qualquer estado  $s$ ,  $\mathcal{A}[a[y \mapsto a_0]]_s = \mathcal{A}[a](s[y \mapsto \mathcal{A}[a_0]_s])$

**Prova:** por indução na estrutura de  $a$ .

**Exercício:**

Defina e prove resultados similares para expressões booleanas.

28