Semântica Denotacional

Linguagens de Programação 2019.2020

Teresa Gonçalves
tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

Significado de um programa

Composicionalidade

Denotação de uma expressão

Denotação de programas while

Semântica denotacional e funções parciais

Semântica Denotacional

História

Christopher Strachey e Dana Scott, final dos anos 60

O que é?

Semântica matemática para programas imperativos

Objetivo

Definir o **significado** (denotação) de programas de forma matemática

Para que serve?

Verificar a correção de um programa

se o output é correto para qualquer input

Significado de um programa

Denotação

Denotação

Função matemática de estados para estados

Estado

Função matemática que representa os valores em memória num ponto de execução do programa

Exemplo

`x:=0; y:=0; while x≤z do (y:=y+x; x:=x+1)`

A denotação é uma função que associa

- estado inicial
  - z é um inteiro não negativo n
- estado final
  - x=z
  - y é a soma de todos os inteiros até n
  - todos as outras posições de memória ficam inalteradas

Composicionalidade

Composicionalidade

O que é?

Princípio da semântica denotacional

O que diz?

O significado de um programa é definido a partir do significado das suas “partes”

A denotação de uma instrução deve ser detalhada o suficiente para capturar **tudo o que é relevante** para o seu comportamento em programas maiores

Utilidade

Permite perceber e raciocinar sobre

- Transformação de programas
- Otimização

Exemplo

`if B then P else Q`

A denotação deve ser explicada apenas a partir das denotações de B, P e Q

Se

B, P e Q têm a mesma denotação que B', P' e Q', respetivamente

Então

`if B' then P' else Q'` possui a mesma denotação que `if B then P else Q`

Denotação de uma expressão

Denotação de uma expressão

O significado de uma expressão é dado através da sua árvore de sintaxe abstrata

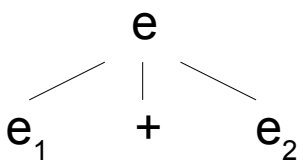
Obtido por indução na estrutura das árvores

Convenção

[[e]] → árvore sintática para a expressão e

Exemplo

[[e1+e2]] corresponde à arvore



SD para números binários

Gramática

```
n ::= b | nb
b ::= 0 | 1
```

Função avaliação

```
A: expressão → número
```

A[[e]] : valor da expressão e

Denotação

```
A[[0]] = 0
A[[1]] = 1
A[[nb]] = A[[n]]*2 + A[[b]]
```

SD para expressões e variáveis

Gramática

```
e ::= v | n | e+e | e-e
n ::= d | nd
d ::= 0|1|2|3|4|5|6|7|8|9
v ::= x | y | z |...
```

Variável

O valor depende do **estado da máquina**

Função avaliação

```
A: expressao*estado → número
```

A[[e]](s) : valor da expressão e no estado s

SD para expressões e variáveis (continuação)

Denotação

```
A[[0]](s) = 0
...
A[[9]](s) = 9
A[[nd]](s) = A[[n]](s)*10 + A[[d]](s)
A[[e1+e2]](s) = A[[e1]](s) + A[[e2]](s)
A[[e1-e2]](s) = A[[e1]](s) - A[[e2]](s)
A[[x]](s) = s(x)
```

Exemplo

```
A[[x+12]](s) = A[[x]](s) + A[[12]]
```

O valor final depende do estado s

s é necessário para avaliar o valor da variável!

Programa while

Programa while

O que é?

Linguagem definida sobre expressões booleanas e de valores

Gramática

```
P ::= x:= e
      | P;P
      | if e then P else P
      | while e do P
```

Qual o objetivo?

modificar valores de variáveis

Exemplo

```
x:=0; y:=0; while x≤z do (y:=y+x; x:=x+1)
```

input:z, output:y

Definições

Denotação de um programa

- Função de programa para instrução

  `I: Programa → Instrução`

- Obtida por indução na estrutura das árvores

  - [[P]] é a árvore sintática do programa P

Definições (continuação)

Denotação de instrução

- Função de estado para estado

  `Instrução: Estado → Estado`

- Função **parcial**
- Baseia-se na função básica de modificar valores

  `modificar(s,x,a) = λv ∈ variáveis.se v=x então a senão s(v)`

  - Obtém-se um estado igual a s, exceto para x (que fica com o valor a)

Denotação de estado

- Função de variáveis para valores

  `Estado: Variáveis → Valores`

- Função **total**

Denotação de um programa

```
I[[x:=e]](s) = modificar(s,x,A[[e]](s))

I[[P1;P2]](s) = I[[P2]](I[[P1]](s))

I[[if e then P1 else P2]](s) =
  se A[[e]](s) então I[[P1]](s)
  senão I[[P2]](s)

I[[while e do P]](s) =
  se não A[[e]](s) então s
  senão I[[while e do P]](I[[P]](s))
```

Semântica da atribuição

`I[[x:=e]](s) = modificar(s,x,A[[e]](s))`

Função estado → estado

s → s'

s' é similar a s, mas x tem o valor de e no estado s

s'(x) = A[[e]](s)

Exemplo

x := 1

estado inicial s_0, com $s_0(x)=4$

$I[[x:=1]](s_0) = s_1$

$s_1 = \text{modificar}(s_0, x, A[[1]](s_0)) = \text{modificar}(s_0, x, 1)$

estado s_0

x	4
...	

estado s_1

x	1
...	

Semântica da composição

$I[[P_1;P_2]](s) = I[[P_2]](I[[P_1]](s))$

Função estado → estado

s → s'

s' é o estado obtido pela aplicação da semântica de P_2 ao estado resultante da aplicação da semântica de P_1 ao estado s

Exemplo

x:=1; x:=x+1

estado inicial s_0, com $s_0(x)=4$

$I[[x:=x+1]](I[[x:=1]](s_0)) = s_2$

$I[[x:=1]](s_0)=s_1$

$s_1 = modificar(s_0,x,A[[1]](s_0)) = modificar(s_0,x,1)$

$I[[x:=x+1]](s_1)=s_2$

$s_2 = modificar(s_1,x,A[[x+1]](s_1)) = modificar(s_1,x,2)$

estado s_0

x	4
...	

estado s_1

x	1
...	

estado s_2

x	2
...	

Semântica da condicional

$$I[[\text{if } e \text{ then } P_1 \text{ else } P_2]](s) =$$
$$\quad \text{se } A[[e]](s) \text{ então } I[[P_1]](s)$$
$$\quad \text{senão } I[[P_2]](s)$$

Função estado → estado

s → s'

s' é o estado obtido pela aplicação no estado s da semântica de

P_1 se A[[e]](s) for verdade ou

P_2 se A[[e]](s) for falso

Exemplo

if x>0 then x:=x+1 else x:=x-1

estado inicial s_0, com $s_0(x)=4$

I[[if x>0 then x:=x+1 else x:=x-1]](s_0)

se A[[x>0]](s_0)

então I[[x:=x+1]](s_0)

senão I[[x:=x-1]](s_0) =

= I[[x:=x+1]](s_0) = s_1

s_1 = modificar(s_0,x,A[[x+1]](s_0)) = modificar(s_0,x,5)

estado s_0

x	4
...	

estado s_1

x	5
...	

Semântica da iteração

```
I[[while e do P]](s) =
   se não A[[e]](s) então s
   senão I[[while e do P]](I[[P]](s))
```

Função recursiva f: estado → estado

`f(s)=s` se `A[[e]]` é falso

`f(s)=f(I[[P]](s))` se `A[[e]]` é verdade

Exemplo

while x>0 do x:=x-1

estado inicial s_0, com $s_0(x)=1$

I[[while x>0 do x:=x-1]](s_0)= s_2

se não A[[x>0]](s_0) então s_0 senão I[[while x>0 do x:=x-1]](s_0)
=

I[[while x>0 do x:=x-1]](I[[x:=x-1]](s_0))

s_1 = I[[x:=x-1]](s_0)

= modificar(s_0,x,A[[x-1]](s_0))

= modificar(s_0,x,0)

s_2 = I[[while x>0 do x:=x-1]](s_1)

= se não A[[x>0]](s_1) então s_1 senão I[[while x>0 do x:=x-1]]
(I[[x:=x-1]](s_1))

Como A[[x>0]](s_1) é falso, tem-se s_2=s_1

Exemplo (continuação)

estado s_0	
x	1
...	

estado s_1	
x	0
...	

estado s_2	
x	0
...	

Qual a semântica do programa

`x:=0; y:=0; while x≤z do (y:=y+x; x:=x+1)` com estado inicial s_0 e $s_0(z)=2$

$s_0=\{x=?, y=?, z=2\}$

Consideremos

s_1 = `I[[ x:=0 ]](`s_0`)`

s_1= `modificar(`s_0`,x,0)`

$s_1=\{x=0, y=?, z=2\}$

s_2 = `I[[ y:=0 ]](`s_1`)`

s_2= `modificar(`s_1`,y,0)`

$s_2=\{x=0, y=0, z=2\}$

Semântica de um programa (continuação)

$I[[x:=0;y:=0;\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](s_0) =$
$= I[[y:=0;\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](s_1) =$
$= I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](s_2) =$

$= \text{se não } A[[x \leq z]](s2) \text{ então } s_2 \text{ senão}$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](I[[y:=y+x;\ x:=x+1]](s_2)) =$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](s_4) =$
$= \text{se não } A[[x \leq z]](s_4) \text{ então } s_4 \text{ senão}$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](I[[y:=y+x;\ x:=x+1]](s_4)) =$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](s_6) =$
$= \text{se não } A[[x \leq z]](s_6) \text{ então } s_6 \text{ senão}$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](I[[y:=y+x;\ x:=x+1]](s_6)) =$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](s_8) =$
$= \text{se não } A[[x \leq z]](s_8) \text{ então } s_8 \text{ senão}$
$I[[\text{while } x \leq z \text{ do } (y:=y+x;\ x:=x+1)]](I[[y:=y+x;\ x:=x+1]](s_8)) = s_8$

Semântica de um programa (continuação)

$s_3 = I[[y:=y+x]](s_2) = \text{modificar}(s_2, y, 0)$

$s_4 = I[[x:=x+1]](s_3) = \text{modificar}(s_3, x, 1)$

$s_5 = I[[y:=y+x]](s_4) = \text{modificar}(s_4, y, 1)$

$s_6 = I[[x:=x+1]](s_5) = \text{modificar}(s_5, x, 2)$

$s_7 = I[[y:=y+x]](s_6) = \text{modificar}(s_6, y, 3)$

$s_8 = I[[x:=x+1]](s_7) = \text{modificar}(s_7, x, 3)$

$s_3 = \{x=0, y=0, z=2\}$

$s_4 = \{x=1, y=0, z=2\}$

$s_5 = \{x=1, y=1, z=2\}$

$s_6 = \{x=2, y=1, z=2\}$

$s_7 = \{x=2, y=3, z=2\}$

$s_8 = \{x=3, y=3, z=2\}$

SD e funções parciais

SD e funções parciais

A SD associa, de forma não ambígua, um programa a uma função parcial de estados para estados

Exemplos

Função parcial não definida em nenhum estado

I[[`while x=x do x:=x`]]

Para qualquer estado s, I[[`while x=x do x:=x`]] não está definido

Função parcial que não está definida para x=y

I[[`while x=y do x:=x`]](s)

É s se x<>y; caso contrário não está definida