

DIM0436

10. Semântica I

Richard Bonichon

20140821

Sumário

- 1 Introdução
- 2 A linguagem While
- 3 Semântica natural
- 4 Semântica operacional estrutural

- 1 Introdução
- 2 A linguagem While
- 3 Semântica natural
- 4 Semântica operacional estrutural

Porque semântica ?

Tem uma sintaxe. Muito bem! O que ela significa ?

- A semântica detalha o **significado** das coisas (i.e. dos programas)
- A formalização da semântica permite interpretar os programas usando matemática.

Em breve

Semântica usa-se:

- 1 para entender uma linguagem de programação
 - 1 Em que podemos confiar como programadores
 - 2 O que devemos fornecer como autor de compilador
- 2 como **ferramenta** no desenho de linguagem
 - 1 para expressar escolhas no desenho, descrever como os diferentes aspectos da linguagem funcionam (juntos)
 - 2 demonstrar propriedades da linguagem (segurança, correção)
- 3 como fundamento para demonstrar propriedades de programas

Semânticas operacionais

- (Plotkin, 1981) e (McCarty, 1960)
- O que o programa calcula
- Descreve o processo computacional dos programas
- Abordagem concreta
- Perto da sintaxe
- Muito usado na construção de [interpretadores](#)

Semântica denotacional

- (Scott-Strachey, 1971)
- O que o programa calcula
- Significado abstrato e matemático dos programas
- Mapeamento de programas em estruturas matemáticas (domínios)
- Não tem necessariamente referências a sintaxe ou a avaliação dela.

Semântica axiomática

- (Floyd, 1967), (Hoare 1969)
- As propriedades estabelecidas por um programa
- Efeitos das instruções sobre o estado do programa por **asserções**
- Asserções são elementos lógicos

Qual é o resultado?

```
#include <stdio.h>
#include <math.h>

int main() {
    double x = 3.2;
    double y = -0.0;
    double z = 0.0;

    printf ("%f\n", x / y);
    printf ("%f\n", x / z);
    printf ("%f\n", z / z);
    printf ("%f\n", sqrt(-1));
}
```

Resposta

Clang

```
1 clang -O1 -lm ieee754.c  
2 ./a.out
```

```
-inf  
inf  
nan  
-nan
```

Clang

```
1 clang -O3 -lm ieee754.c  
2 ./a.out
```

```
-inf  
inf  
nan  
-nan
```

Outros exemplos

```
1  #include <stdio.h>
2  int main(void){
3  int* p;
4  {
5      int x = 3 ;
6      p = &x;
7  }
8
9  printf("%d\n", *p);
10 return 1;
11 }
```

```
1  #include <stdio.h>
2
3  int main() {
4      char *p = "foobar";
5      p[0] = 'F';
6
7      printf("%s\n", p);
8      return 0;
9  }
```

```
1  #include <stdio.h>
2
3  int main() {
4      int a[] = { 0 , 1, 2, 4 };
5      int i = 1;
6      int j;
7      a[i] = i++;
8
9      for (j = 0; j < 4; j++) {
10         printf("a[%d] = %d\n", j, a[j]);
11     }
12
13     return 0;
14 }
```

- 1 Introdução
- 2 A linguagem While
- 3 Semântica natural
- 4 Semântica operacional estrutural

Sintaxe

Categorias sintáticas

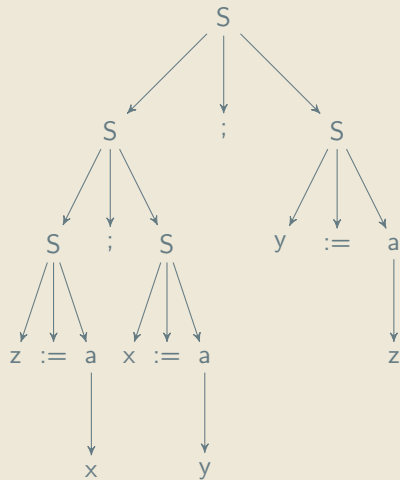
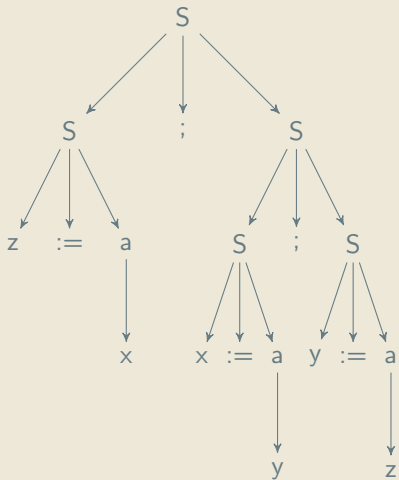
- n, n_i, n' = elementos numéricos (Num)
- x = variáveis (Var)
- a = expressões aritméticas (\mathcal{A}_{exp})
- b = expressões booleanas (\mathcal{B}_{exp})
- S = instruções

BNF

$$\begin{aligned} a & ::= n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2 \\ b & ::= \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\ S & ::= x := a \mid \text{skip} \mid S_1; S_2 \\ & \quad \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \\ & \quad \mid \text{while } b \text{ do } S \end{aligned}$$

Árvore de sintaxe abstrata

Árvores de sintaxe abstratas par $z := x; x := y; y := z$



Semântica dos números

Números binários

$$n := 0 \mid 1 \mid n0 \mid n1$$

Função semântica

Para determinar o valor numérico de um elemento de n

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

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

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

$$\mathcal{N}[[n0]] = 2 * \mathcal{N}[[n]]$$

$$\mathcal{N}[[n1]] = 1 + 2 * \mathcal{N}[[n]]$$

Semântica das expressões

Funções semânticas

- A função $s : \text{Var} \rightarrow \mathbb{Z}$ dá o valor duma variável num estado do programa.
- Precisamos da função

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

Definição (\mathcal{A})

$$\mathcal{A}[[n]]s = \mathcal{N}[[n]]$$

$$\mathcal{A}[[x]]s = s(x)$$

$$\mathcal{A}[[a_1 + a_2]]s = \mathcal{A}[[a_1]]s + \mathcal{A}[[a_2]]s$$

$$\mathcal{A}[[a_2 * a_1]]s = \mathcal{A}[[a_1]]s * \mathcal{A}[[a_2]]s$$

$$\mathcal{A}[[a_1 - a_2]]s = \mathcal{A}[[a_1]]s - \mathcal{A}[[a_2]]s$$

Exemplo

Seja $s(x) = 5$

$$\begin{aligned}\mathcal{A}[[x + 1]]_s &= \mathcal{A}[[x]]s + \mathcal{A}[[1]]s \\ &= s(x) + \mathcal{N}[[1]] \\ &= 5 + 1 \\ &= 5\end{aligned}$$

Expressões booleanas

Definição

$$\mathcal{B}[\text{true}]s = \top$$

$$\mathcal{B}[\text{false}]s = \perp$$

$$\mathcal{B}[a_1 = a_2]s = \begin{cases} \top & \text{se } \mathcal{A}[a_1]s = \mathcal{A}[a_2]s \\ \perp & \text{se } \mathcal{A}[a_1]s \neq \mathcal{A}[a_2]s \end{cases}$$

$$\mathcal{B}[a_1 \leq a_2]s = \begin{cases} \top & \text{se } \mathcal{A}[a_1]s \leq \mathcal{A}[a_2]s \\ \perp & \text{se } \mathcal{A}[a_1]s > \mathcal{A}[a_2]s \end{cases}$$

$$\mathcal{B}[\neg b]s = \begin{cases} \top & \text{se } \mathcal{B}[b]s = \perp \\ \perp & \text{se } \mathcal{B}[b]s = \top \end{cases}$$

$$\mathcal{B}[b_1 \wedge b_2]s = \begin{cases} \top & \text{se } \mathcal{B}[b_1]s = \top \text{ e } \mathcal{B}[b_2]s = \top \\ \perp & \text{se } \mathcal{B}[b_1]s = \perp \text{ ou } \mathcal{B}[b_2]s = \perp \end{cases}$$

Exercícios

Suponha $s(x) = 7$, determine $\mathcal{B}[\neg(x = 1)]s$.

Suponha

$$\begin{aligned} b ::= & \text{ true } \mid \text{ false } \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\ & \mid a_1 \neq a_2 \mid a_1 \geq a_2 \mid a_1 < a_2 \mid a_1 > a_2 \\ & \mid b_1 \vee b_2 \mid b_1 \Rightarrow b_2 \mid b_1 \Leftrightarrow b_2 \end{aligned}$$

Dar uma extensão composicional da função semântica \mathcal{B} .

Variáveis livres e propriedade

Variáveis livres numa expressão aritmética

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

Teorema

Seja s e s' dois estados tal que $s(x) = s'(x)$, $\forall x \in FV(a)$. Então $\mathcal{A}[a]_s = \mathcal{A}[a]_{s'}$

Proof.

Por indução sobre a . □

Substituições

Expressões aritméticas

$$n[y \mapsto a_0] = n$$

$$x[y \mapsto a_0] = \begin{cases} a_0 & \text{se } x = y \\ x & \text{se } x \neq y \end{cases}$$

$$(a_1 \circ a_2)[y \mapsto a_0] = a_1[y \mapsto a_0] + a_2[y \mapsto a_0] \quad \circ \in \{+, -, *\}$$

Estados

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

Exercícios

- 1 Seja s e s' dois estados tal que $s(x) = s'(x), \forall x \in FV(b)$. Mostre que $\mathcal{B}[b]s = \mathcal{B}[b]s'$
- 2 Mostre que $\mathcal{A}[a[y \mapsto a_0]]s = \mathcal{A}[a](s[y \mapsto \mathcal{A}[a_0]s])$
- 3 Defina a substituição para expressões booleanas $b[y \mapsto a_0]$ como a expressão booleana b com todas as ocorrências da variável y substituída pela expressão aritmética a_0 . Mostre que sua definição satisfaz

$$\mathcal{B}[a[y \mapsto a_0]]s = \mathcal{B}[a](s[y \mapsto \mathcal{A}[a_0]s])$$

- 1 Introdução
- 2 A linguagem While
- 3 Semântica natural**
- 4 Semântica operacional estrutural

Introdução

Objetivo

- É uma semântica **operacional**
- Descreve como o resultado **geral** é obtido.
- Relação entre o estado **inicial** e o estado **final**

Notação

- $\langle S, s \rangle$ representa o comando S a ser executado no estado s
- s representa um estado terminal (final)

Forma das regras

$$\frac{\langle S_1, s_1 \rangle \rightarrow s'_1 \quad \dots \quad \langle S_n, s_n \rangle \rightarrow s'_n}{\langle S, s \rangle \rightarrow s''}$$

Regras

$$\frac{}{\langle x := a, s \rangle \rightarrow s[x \mapsto \mathcal{A}[[a]]s]} \text{Assigns}$$

$$\frac{}{\langle \text{skip}, s \rangle \rightarrow s} \text{Skip}$$

$$\frac{\langle S_1, s \rangle \rightarrow s' \quad \langle S_2, s' \rangle \rightarrow s''}{\langle S_1; S_2, s \rangle \rightarrow s''} \text{Comp}$$

$$\frac{\langle S_1, s \rangle \rightarrow s' \quad \mathcal{B}[[b]]s = \top}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \text{If}_{\top}$$

$$\frac{\langle S_2, s \rangle \rightarrow s' \quad \mathcal{B}[[b]]s = \perp}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \text{If}_{\perp}$$

$$\frac{\langle S, s \rangle \rightarrow s' \quad \langle \text{while } b \text{ do } S, s' \rangle \rightarrow s'' \quad \mathcal{B}[[b]]s = \top}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \text{While}_{\top}$$

$$\frac{\mathcal{B}[[b]]s = \perp}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s} \text{While}_{\perp}$$

Exercícios

$$\text{Suponha } s_0(a) = \begin{cases} 5 & \text{se } a = x \\ 7 & \text{se } a = y \\ 0 & \text{senão} \end{cases}$$

Escreva a árvore de derivação de $\langle (z := x; x := y;)y := z, s_0 \rangle$

Seja s um estado tal que $s(x) = 3$. Mostre que que

$$\langle x := 1; \text{while } \neg(x = 1) \text{ do } (y := y * x; x := x + 1), s \rangle \rightarrow s[y \mapsto 6][x \mapsto 1]$$

Terminação

Definição (Vocabulário)

A execução de um comando S no estado s

- **termina** sse existe um estado s' tal que $\langle S, s \rangle \rightarrow s'$
- **sempre termina** sse termina para todo estado s
- **não termina** sse não existe um estado s' tal que $\langle S, s \rangle \rightarrow s'$
- **nunca termina** sse não termina para todo estado s

Exercício

Determine se os comando seguintes sempre terminam ou não, sempre rodam ou não.

- *while* $\neg(x = 1)$ *do* ($y := y * x; x := x - 1$)
- *while* $1 \leq x$ *do* ($y := y * x; x := x - 1$)
- *while* *true do skip*

Equivalência semântica

Definição

Dois comandos S_1 e S_2 são **semanticamente equivalentes** se

$$\forall s, s' \langle S_1, s \rangle \rightarrow s' \text{ sse } \langle S_2, s \rangle \rightarrow s'$$

Teorema

O comando

while b do S

é semanticamente equivalente a

if b then (S; while b do S) else skip

Exercícios

Repeat

Estender a linguagem **While** com o comando

`repeat S until b`

- Definir a relação \rightarrow para este comando
- Mostre que `repeat S until b e S; if b then skip else (repeat S until b)` são semanticamente equivalente

For

Estender a linguagem **While** com o comando

`for $x := a_1$ to a_2 do S`

- Definir a relação \rightarrow para este comando
- Avaliar `$y := 1; for z := 1 to x do (y := y * x, x := x - 1)$` num estado s onde $s(x) = 5$

Determinismo

Definição

$$\langle S, s \rangle \rightarrow s' \wedge \langle S, s \rangle \rightarrow s'' \Rightarrow s' = s''$$

Proof.

Por indução. □

Função semântica : \mathcal{S}_{NS}

Definição

A semântica dos comandos é um função (parcial):

$$\mathcal{S}_{NS} : \text{Stm} \rightarrow \text{State} \rightarrow \text{State}$$

Ou seja:

$$\mathcal{S}_{NS}[[S]] \in \text{State} \rightarrow \text{State}$$

Definição

$$\mathcal{S}_{NS}[[S]]s = \begin{cases} s' & \text{se } \langle S, s \rangle \rightarrow s' \\ \emptyset & \text{senão} \end{cases}$$

Exemplo

$\mathcal{S}_{NS}[[\text{while true do skip}]]s = ??$

Semântica natural para expressões

Definir uma semântica natural para

- 1 expressões aritméticas
- 2 expressões booleanas

- 1 Introdução
- 2 A linguagem While
- 3 Semântica natural
- 4 Semântica operacional estrutural

Introdução

Objetivo

- Semântica operacional
- Descreve como as etapas individuais são obtidas.

Notações

A relação de transição é da forma:

$$\langle S, s \rangle \leftrightarrow \gamma$$

- se $\gamma = \langle S', s' \rangle$, a execução de S a partir de s não é **completa**
- se $\gamma = s'$, a execução é **terminada** e o estado final é s' .

Semântica operacional estrutural

$$\frac{}{\langle x := a, s \rangle \hookrightarrow s[x \mapsto \mathcal{A}[[a]]s]} \text{ Assigns}$$

$$\frac{}{\langle \text{skip}, s \rangle \hookrightarrow s} \text{ Skip}$$

$$\frac{\langle S_1, s \rangle \hookrightarrow \langle S'_1, s' \rangle}{\langle S_1; S_2, s \rangle \hookrightarrow \langle S'_1; S_2, s' \rangle} \text{ Comp}_1$$

$$\frac{\langle S_1, s \rangle \hookrightarrow s'}{\langle S_1; S_2, s \rangle \hookrightarrow \langle S_2, s' \rangle} \text{ Comp}_2$$

$$\frac{\mathcal{B}[[b]]s = \top}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \hookrightarrow \langle S_1, s \rangle} \text{ If}_\top$$

$$\frac{\mathcal{B}[[b]]s = \perp}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \hookrightarrow \langle S_2, s \rangle} \text{ If}_\perp$$

$$\frac{}{\langle \text{while } b \text{ do } S, s' \rangle \hookrightarrow \langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, s \rangle} \text{ While}$$

Derivações

Uma **sequência de derivação** do comando S a partir do estado s é:

- uma sequência **finita**

$$\gamma_0, \gamma_1, \gamma_2, \dots, \gamma_n$$

de configurações tal que

- ▶ $\gamma_0 = \langle S, s \rangle$,
- ▶ $\gamma_i \hookrightarrow \gamma_{i+1}$ para $0 \leq i \leq n, n \geq 0$,
- ▶ γ_n é uma configuração terminal ou presa.

- uma sequência **infinita**

$$\gamma_0, \gamma_1, \gamma_2, \dots$$

de configurações tal que

- ▶ $\gamma_0 = \langle S, s \rangle$,
- ▶ $\gamma_i \hookrightarrow \gamma_{i+1}$ para $0 \leq i$,

Exemplo

Suponha $s_0(a) = \begin{cases} 5 & \text{se } a = x \\ 7 & \text{se } a = y \\ 0 & \text{senão} \end{cases}$

A seqüência de derivação $\langle z := x; x := y; y := z, s_0 \rangle$ é

$$\begin{aligned} \langle (z := x; x := y); y := z, s_0 \rangle &\hookrightarrow \langle x := y; y := z, s_0[z \mapsto 5] \rangle \\ &\hookrightarrow \langle y := z, (s_0[z \mapsto 5])[x \mapsto 5] \rangle \\ &\hookrightarrow \langle (s_0[z \mapsto 5])[x \mapsto 5][y \mapsto 5] \rangle \end{aligned}$$

$$\frac{\langle z := x, s_0 \rangle \hookrightarrow s_0[z \mapsto 5]}{\langle z := x; x := y, s_0 \rangle \hookrightarrow \langle x := y, s_0[z \mapsto 5] \rangle}$$

$$\frac{\langle z := x; x := y, s_0 \rangle \hookrightarrow \langle x := y, s_0[z \mapsto 5] \rangle}{\langle (z := x; x := y); y := z, s_0 \rangle \hookrightarrow \langle x := y; y := z, s_0[z \mapsto 5] \rangle}$$

$$\langle (z := x; x := y); y := z, s_0 \rangle \hookrightarrow \langle x := y; y := z, s_0[z \mapsto 5] \rangle$$

Exercícios

Seja s um estado tal que $s(x) = 3$. Escreva a sequência de derivação do trecho seguinte:

$$\langle x := 1; \text{while } \neg(x = 1) \text{ do } (y := y * x; x := x + 1), s \rangle$$

Construa uma sequência de derivação para o trecho seguinte:

$$z := 0; \text{while } y \leq x \text{ do } (z := z + 1; x := x - y)$$

onde

- $s(x) = 17$
- $s(y) = 5$

Exercícios

Repeat

Estender a linguagem **While** com o comando

`repeat S until b`

- Especifica a semântica operacional estrutural para este comando

For

Estender a linguagem **While** com o comando

`for $x := a_1$ to a_2 do S`

- Especifica a semântica operacional estrutural para este comando

Terminação

Definição (Vocabulário)

A execução de um comando S no estado s

- **termina (com sucesso)** sse existe uma sequência de derivação **finita** a partir de $\langle S, s \rangle$
- **sempre termina** sse termina para todo estado s
- **não termina** sse existe uma sequência de derivação **infinita** a partir de $\langle S, s \rangle$
- **nunca termina** sse não termina para todo estado s

Propriedade

Teorema

Se $\langle S_1; S_2, s \rangle \hookrightarrow^k s''$, existe um estado s' e números naturais k_1 e k_2 tal que

- $\langle S_1, s \rangle \hookrightarrow^{k_1} s'$ e
- $\langle S_2, s' \rangle \hookrightarrow^{k_2} s''$,

onde $k = k_1 + k_2$

Proof.

Por indução sobre k . □

A semântica dos comandos é a função (parcial)

$$\mathcal{S}_{SOS} : \text{Stm} \rightarrow \text{State} \rightarrow \text{State}$$

Definição

$$\mathcal{S}_{SOS} \llbracket S \rrbracket s = \begin{cases} s' & \text{se } \langle S, s \rangle \hookrightarrow^* s' \\ \emptyset & \text{senão} \end{cases}$$

Equivalência das semânticas

Teorema

Para todo comando S de While, $\mathcal{S}_{NS}[[S]] = \mathcal{S}_{SOS}[[S]]$

- Se a execução de S a partir de um estado terminar numa semântica, ela termina também na outra com o mesmo estado.
- Se a execução de S a partir de um estado **não** terminar numa semântica, ela não termina também na outra.

Demonstração

Para qualquer comando S de While, qualquer estados s, s' :

$$\langle S, s \rangle \rightarrow s' \Rightarrow \langle S, s \rangle \hookrightarrow^* s'$$

* Para qualquer comando S de While, qualquer estados s, s' :

$$\langle S, s \rangle \hookrightarrow^k s' \Rightarrow \langle S, s \rangle \rightarrow s'$$



Proof.

Por indução sobre k . □

Resumo

- 1 Introdução
- 2 A linguagem While
- 3 Semântica natural
- 4 Semântica operacional estrutural

Referências

-  Hanne Riis Nielson and Flemming Nielson, *Semantics with applications: An appetizer*, Undergraduate Topics in Computer Science, Springer, 2007.
-  Flemming Nielson, Hanne Riis Nielson, and Chris Hankin, *Principles of program analysis (2. corr. print)*, Springer, 2005.

Perguntas ?



<http://dimap.ufrn.br/~richard/dim0436>