

Semântica Operacional

31

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
```

33

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)

32

Semântica Operacional

- A semântica já dada às expressões aritméticas e booleanas apenas *consultam* o estado.
- O papel de um comando da linguagem é alterar o estado. A semântica dos comandos irá portanto *modificar* o estado.
- A semântica operacional descreve *como* os programas são executados e não apenas nos resultados da sua execução.

34

Semântica Operacional

Duas abordagens:

- **Semântica Natural** (*Semântica de Avaliação* ou *Big-step*)

O seu propósito é descrever como os *resultados globais* das execuções são obtidos.

- **Semântica Operacional Estrutural** (*Semântica de Transições* ou *Small-step*)

O seu propósito é descrever como os *passos individuais* das computações são executados.

35

Sistemas de Transição

Na semântica operacional o significado de um comando é dado por um sistema de transição.

Um *sistema de transição* é constituído por um conjunto de *configurações* e por uma *relação binária de transição* entre configurações. Há dois tipos de configurações:

$\langle S, s \rangle$ indica que o comando S é para ser executado no estado s .

s representa um estado final (é uma *configuração terminal*)

A *relação de transição* descreve como a execução é feita. A diferença entre a abordagem *big-step* e *small-step* está na forma como a relação de transição é especificada.

36

Semântica Natural

37

Semântica Natural (de Avaliação)

- A relação de transição → especifica, para cada comando, a relação entre o estado inicial e o estado final.

$$\langle S, s \rangle \rightarrow s'$$

- → é definida indutivamente por um conjunto de *regras*

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

premissas *conclusão* *condições*

As regras sem premissas chamam-se *axiomas*.

38

Regras de Avaliação

$$\begin{array}{ll}
 [\text{ass}_{\text{ns}}] & \langle x := a, s \rangle \rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s] \\
 [\text{skip}_{\text{ns}}] & \langle \text{skip}, s \rangle \rightarrow s \\
 [\text{comp}_{\text{ns}}] & \frac{\langle S_1, s \rangle \rightarrow s', \langle S_2, s' \rangle \rightarrow s''}{\langle S_1; S_2, s \rangle \rightarrow s''}
 \end{array}$$

39

Regras de Avaliação

$$\begin{array}{ll}
 [\text{if}_{\text{ns}}^{\text{tt}}] & \frac{\langle S_1, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \text{ if } \mathcal{B}[\![b]\!]s = \text{tt} \\
 [\text{if}_{\text{ns}}^{\text{ff}}] & \frac{\langle S_2, s \rangle \rightarrow s'}{\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \rightarrow s'} \text{ if } \mathcal{B}[\![b]\!]s = \text{ff}
 \end{array}$$

40

Regras de Avaliação

$$\begin{array}{ll}
 [\text{while}_{\text{ns}}^{\text{tt}}] & \frac{\langle S, s \rangle \rightarrow s', \langle \text{while } b \text{ do } S, s' \rangle \rightarrow s''}{\langle \text{while } b \text{ do } S, s \rangle \rightarrow s''} \text{ if } \mathcal{B}[\![b]\!]s = \text{tt} \\
 [\text{while}_{\text{ns}}^{\text{ff}}] & \langle \text{while } b \text{ do } S, s \rangle \rightarrow s \text{ if } \mathcal{B}[\![b]\!]s = \text{ff}
 \end{array}$$

41

Árvores de Derivação

- As transições são *derivadas* construindo *árvores de derivação*.
- A *raiz* da derivação é a transição que queremos derivar e as *folhas* são instâncias dos axiomas.
- Os *nodos internos* são conclusões de instâncias de regras.
- As condições instanciadas das regras têm que ser satisfeitas.

42

Exercício

- Defina na linguagem **While** os seguintes programas:
 - SWAP — Troca de valores entre as variáveis x e y.
 - MIN — Cálculo em m do mínimo de x, y e z.
 - EXP — Cálculo em r do valor de x elevado a y.
- Seja s o estado que mapeia todas as variáveis em 0, excepto x e y. Nestes casos: s x = 3 e s y = 2.

Construa árvores de derivação para as transições correspondentes à execução de cada um dos programas nos estado s.

43

Terminação

- A execução de um programa S num estado s
 - termina* se e só se existir um estado s' tal que $\langle S, s \rangle \rightarrow s'$
 - diverge* (ou *entra em ciclo*) se e só se não existir nenhum estado s' tal que $\langle S, s \rangle \rightarrow s'$

Exercício: Como é que se comportam os seguintes programas?

- `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`

44

Equivalência Semântica

O sistema de transição permite-nos argumentar acerca dos programas e das suas propriedades.

Definição: Dois programas S_1 e S_2 dizem-se *semanticamente equivalentes* se, para todos os estados s e s' ,

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

Exercício: Prove que os programas `while b do S` e `if b then (S; while b do S) else skip` são semanticamente equivalentes.

45

Indução na estrutura da derivação

Provar uma propriedade para todas as árvores de derivação.

Casos de base (os axiomas)

- Provar que a propriedade se verifica para todos os axiomas.

Casos indutivos (as regras)

- Para cada regra assumir que a propriedade se verifica para as premissas da regra (são as *hipóteses de indução*) e provar que a propriedade também se verifica para a conclusão, desde que as condições da regra sejam satisfeitas.

46

Determinismo

A semântica natural aqui apresentada é **determinista**.

Teorema: Para quaisquer estados s, s', s'' e programa S ,
se $\langle S, s \rangle \rightarrow s'$ e $\langle S, s \rangle \rightarrow s''$ então $s' = s''$.

Prova: Por indução na estrutura da derivação de $\langle S, s \rangle \rightarrow s'$

Assume-se que $\langle S, s \rangle \rightarrow s'$ e demonstra-se que
se $\langle S, s \rangle \rightarrow s''$ então $s' = s''$.

47

A função semântica \mathcal{S}_{ns}

O significado de um programa pode ser visto como uma *função parcial* de **State** para **State**.

Definição: $\mathcal{S}_{\text{ns}}: \text{Stm} \rightarrow (\text{State} \hookrightarrow \text{State})$

$$\mathcal{S}_{\text{ns}}[S]s = \begin{cases} s' & \text{if } \langle S, s \rangle \rightarrow s' \\ \text{undef} & \text{otherwise} \end{cases}$$

Para qualquer programa S , $\mathcal{S}_{\text{ns}}[S] \in \text{State} \hookrightarrow \text{State}$
é uma função parcial.

Exercício: Qual o valor de $\mathcal{S}_{\text{ns}}[\text{while true do skip}]$?

48

Semântica natural para expressões

- A semântica das expressões aritméticas foi dada pela função semântica \mathcal{A} .
- Podemos ter uma abordagem operacional e definir uma semântica natural para as expressões aritméticas.
- Teremos neste caso dois tipos de configurações:

$\langle a, s \rangle$ denota que a é avaliada no estado s

z denota o valor final (um elemento de \mathbf{Z})

49

Semântica natural para expressões

A relação de transição tem a forma $\langle a, s \rangle \rightarrow_{\text{Aexp}} z$

Exemplo de algumas regras:

$$\langle n, s \rangle \rightarrow_{\text{Aexp}} \mathcal{N}[n]$$

$$\langle x, s \rangle \rightarrow_{\text{Aexp}} s \ x$$

$$\frac{\langle a_1, s \rangle \rightarrow_{\text{Aexp}} z_1, \langle a_2, s \rangle \rightarrow_{\text{Aexp}} z_2}{\langle a_1 + a_2, s \rangle \rightarrow_{\text{Aexp}} z} \quad \text{where } z = z_1 + z_2$$

Exercício: Complete a especificação do sistema de transição e prove que o significado dado por esta definição é o mesmo do que por \mathcal{A} .

Exercício: Defina uma semântica natural para expressões booleanas.

50

Semântica natural para expressões

Exercício: Imagine que pretendemos acrescentar operadores com efeitos laterais (como o `++x` e o `x++` do C) à linguagem de expressões aritméticas.

- Proponha uma semântica natural que permita capturar o efeito da avaliação destas novas expressões.
- Que consequências isto acarreta na semântica dos programas?
- Proponha um conjunto de regras de avaliação que captem corretamente o comportamento destes novos programas.