DIM0436

11. Semântica II

Richard Bonichon

20140828

Sumário

1. Extensões
2. Semântica denotacional

1. Extensões
2. Semântica denotacional

O comando abort

BNF

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

Observações

- A semântica natural não pode distinguir entre laço infinito e terminação anormal
- Na semântica operacional estrutural, laços infinitos são sequências infinitas de derivação e terminação anormal são sequências finitas numa configuração bloqueada.

Exercício

Assunto

Estender `While` com a instrução

`assert b before S`

Se `b` for verdadeira, `S` será executada, senão o programa termina (por exemplo com `abort`).

- Estender a semântica natural de `While`
- Mostre que
  - `assert true before S` é semanticamente equivalente a `S`
  - `assert false before S` não é semanticamente equivalente a `while true do` `skip` nem a `skip`.

Não determinismo

Sintaxe

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

Semânticas

$$\frac{\langle S_1, s\rangle \to s'}{\langle S_1 \text{ or } S_2, s\rangle \to s'}$$

$$\frac{\langle S_2, s\rangle \to s'}{\langle S_1 \text{ or } S_2, s\rangle \to s'}$$

$$\frac{}{\langle S_1 \text{ or } S_2, s\rangle \hookrightarrow \langle S_1, s\rangle}$$

$$\frac{}{\langle S_1 \text{ or } S_2, s\rangle \hookrightarrow \langle S_2, s\rangle}$$

Exercícios

Nos programas abaixo,

1. quantas árvores de derivação são possíveis (semântica natural)?
2. quantas sequências de derivação (semântica operacional estrutural)?

- `<x := 1 or (x := 2; x := x + 2), s>`
- `<(while true do skip) or (x := 2; x := x + 2), s>`

Observações

- Na semântica natural, indeterminismo vai remover laços infinitos, quando possível.
- Na semântica operacional estrutural, indeterminismo não remover laços infinitos.

Paralelismo

Sintaxe

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

Semânticas

Tentativa

$$\frac{\langle S_1, s\rangle \rightarrow s' \qquad \langle S_2, s'\rangle \rightarrow s"}{\langle S_1 \text{ par } S_2, s\rangle \rightarrow s"}$$

$$\frac{\langle S_2, s\rangle \rightarrow s' \qquad \langle S_1, s'\rangle \rightarrow s"}{\langle S_1 \text{ par } S_2, s\rangle \rightarrow s'}$$

$$\frac{\langle S_1, s\rangle \hookrightarrow \langle S_1', s'\rangle}{\langle S_1 \text{ par } S_2, s\rangle \hookrightarrow \langle S_1' \text{ par } S_2, s'\rangle}$$

$$\frac{\langle S_2, s\rangle \hookrightarrow \langle S_2', s'\rangle}{\langle S_1 \text{ par } S_2, s\rangle \hookrightarrow \langle S_1 \text{ par } S_2', s'\rangle}$$

$$\frac{\langle S_1, s\rangle \hookrightarrow s'}{\langle S_1 \text{ par } S_2, s\rangle \hookrightarrow \langle S_2, s'\rangle}$$

$$\frac{\langle S_2, s\rangle \hookrightarrow s'}{\langle S_1 \text{ par } S_2, s\rangle \hookrightarrow \langle S_1, s'\rangle}$$

Observações

- Na semântica natural, a execução dos elementos imediatos é uma entidade atômica: não pode-se expressar a intercalação dos cálculos.
- Na semântica operacional estrutural, o foco nas pequenas etapas do cálculo permite essa expressividade.

Exercício

Protect

Considere uma extensão da linguagem While que também contem a instrução `protect S end`

Especifique uma semântica natural para essa extensão.

Blocos

Sintaxe

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

Semântica dos blocos

Variáveis declaradas

$$\begin{aligned} DV(\text{var } x := a; D_V) &= \{x\} \\ &\quad \cup DV(D_V) \\ DV(\varepsilon) &= \emptyset \end{aligned}$$

Extensão da substituição

$$(s'[X \mapsto s])(x) = \begin{cases} s(x) \text{ se } x \in X \\ s'(x) \text{ se } x \notin X \end{cases}$$

Regras (escopo dinâmico)

$$\frac{\langle D_v, s\rangle \rightarrow_D s' \qquad \langle S, s'\rangle \rightarrow s"}{\langle \text{begin } D_V \ S \text{ end}, s\rangle \rightarrow s"[DV(D_V) \mapsto s]} \text{ block}$$

$$\frac{\langle D_v, s[x \mapsto \mathcal{A}[\![a]\!]s]\rangle \rightarrow_D s'}{\langle \text{var } x := a; D_V, s\rangle \rightarrow_D s'} \text{ var}$$

$$\frac{}{\langle \varepsilon, s\rangle \rightarrow_D s} \text{ none}$$

Rotinas

Sintaxe

$$
\begin{array}{rcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2 \\
b & ::= & \text{true} \mid \text{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & x := a \mid \text{skip} \mid S_1; S_2 \\
 & \mid & \text{if } b \text{ then } S_1 \text{ else} S_2 \\
 & \mid & \text{while } b \text{ do } S \\
 & \mid & \text{begin } D_V D_P \; S \text{ end} \\
 & \mid & \text{call } p \\
D_V & ::= & \text{var } x := a; D_V \mid \varepsilon \\
D_P & ::= & \text{proc } p \text{ is } S; D_P \mid \varepsilon
\end{array}
$$

Questão de escopo

Um programa

```
begin var x := 0;
      proc p is x := x * 2;
      proc q is call p;
      begin var x := 5;
            proc p is
              x := x + 1;
            call q;
            y := x;
     end
end
```

Escopo dinâmico completo

- Qual é o resultado ?

Escopo misto

- Qual é o resultado ?

Escopo estático

- Qual é o resultado ?

Adição de um contexto

Env_P

Env_P : Pname $\rightarrow$ Stm

Transições

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

Regras (escopo dinâmico)

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

$$\frac{}{env_P \vdash \langle \text{skip}, s\rangle \to s}\ \text{Skip}$$

$$\frac{\begin{array}{c} env_P \vdash \langle S_1, s\rangle \to s' \\ env_P \vdash \langle S_2, s'\rangle \to s'' \end{array}}{env_P \vdash \langle S_1; S_2, s\rangle \to s''}\ \text{Comp}$$

$$\frac{env_P \vdash \langle S_1, s\rangle \to s' \qquad \mathcal{B}[\![b]\!]s = \top}{env_P \vdash \langle \text{if } b \text{ then } S_1 \text{ else } S_2, s\rangle \to s'}\ If_\top$$

$$\frac{env_P \vdash \langle S_2, s\rangle \to s' \qquad \mathcal{B}[\![b]\!]s = \bot}{env_P \vdash \langle \text{if } b \text{ then } S_1 \text{ else } S_2, s\rangle \to s'}\ If_\bot$$

$$\frac{env_P \vdash \langle S, s\rangle \to s' \qquad env_P \vdash \langle while\ b\ do\ S, s'\rangle \to s'' \qquad \mathcal{B}[\![b]\!]s = \top}{env_P \vdash \langle while\ b\ do\ S, s\rangle \to s''}\ While_\top$$

$$\frac{\mathcal{B}[\![b]\!]s = \bot}{env_P \vdash \langle while\ b\ do\ S, s\rangle \to s}\ While_\bot$$

Regras para bloco e rotinas

$$\frac{\langle D_v, s\rangle \to_D s' \qquad upd(D_P, env_P) \vdash \langle S, s'\rangle \to s"}{env_P \vdash \langle \text{begin } D_V \; S \text{ end}, s\rangle \to s"[DV(D_V) \mapsto s]} \text{ block}$$

$$\frac{env_P \vdash \langle S, s\rangle \to s'}{env_P \vdash \langle \text{call } p, s\rangle \to s'} \text{ call}$$

com

$$\begin{aligned} upd(\text{proc } p \text{ is } S; D_P, env_P) &= upd(D_P, env_P[p \mapsto S]) \\ upd(\varepsilon, env_P) &= env_P \end{aligned}$$

Exercício

Assunto

Escreva uma árvore de derivação desse programa a partir do estado s, onde $s(x) = 3$.

```
begin proc fac is
  begin
    var z := x;
    if x = 1 then skip
    else (x := x - 1; call fac; y = z * y)
  end
  (y := 1; call fac)
end
```

Escopo estático para rotinas

Extensão do contexto

Env_P : Pname $\to$ Stm $\times$ Env_P

Atualização

$$\begin{array}{rcl} upd(\text{proc } p \text{ is } S; D_P, env_P) & = & upd(D_P, env_P[p \mapsto S, env_P]) \\ upd(\varepsilon, env_P) & = & env_P \end{array}$$

Não-Recursiva

$$\frac{env_P(p) = (S, env'_P) \quad env'_P \vdash \langle S, s\rangle \to s'}{env_P \vdash \langle \text{call } p, s\rangle \to s'}\ \text{Call}$$

Recursiva

$$\frac{env_P(p) = (S, env'_P) \quad env'_P[p \mapsto (S, env'_P)] \vdash \langle S, s\rangle \to s'}{env_P \vdash \langle \text{call } p, s\rangle \to s'}\ \text{Call}$$

1 Extensões

2 Semântica denotacional

Introdução

Objetivo

- Descrever o efeito da execução de um programa, i.e. uma associação entre estados iniciais e finais.

Observações

- Define uma função semântica por categoria sintática
- Descreve mapeamentos de construções sintáticas a objetos matemáticos
- Funções semânticas são composicionais
- As funções $\mathcal{A}$ e $\mathcal{B}$ descrevendo a semântica das expressões aritméticas e booleanas são definições denotacionais.

Definição

Definição

Definimos a função $[\![\cdot]\!] : \text{Stm} \rightarrow \text{State} \rightarrow \text{State}$

$$
\begin{array}{rcl}
[\![x := a]\!]s & = & s[x \mapsto \mathcal{A}[\![a]\!]s] \\
[\![skip]\!] & = & id \\
[\![S_1; S_2]\!] & = & [\![S_1]\!] \circ [\![S_2]\!] \\
[\![\text{if } b \text{ then } S_1 \text{ then } S_2]\!] & = & \text{cond}(\mathcal{B}[\![b]\!]s, [\![S_1]\!], [\![S_2]\!]) \\
[\![\text{while } b \text{ do } S]\!] & = & \text{fix } F
\end{array}
$$

com

- $id(x) = x$
- $Fg = cond(\mathcal{B}[\![b]\!], g \circ [\![S]\!]s, id)$

Observações

Exemplo (Aplicação à sequência)

$$
\begin{array}{rcl}
[\![S_1; S_2]\!]s & = & [\![S_1]\!]s \circ [\![S_2]\!]s \\
& = & \begin{cases} s" & \text{se } \exists s', [\![S_1]\!]s = s' \wedge [\![S_2]\!]s' = s" \\ \oslash & \text{se } [\![S_1]\!]s = \oslash \vee [\![S_1]\!]s = s' \wedge [\![S_2]\!]s' = \oslash \end{cases}
\end{array}
$$

A função cond

Definição

cond : (State $\to$ T) $\times$ (State $\to$ State) $\times$ (State $\to$ State) $\to$ State $\to$ State

$$cond(p, f, g)s = \begin{cases} fs \text{ se } ps = \top \\ gs \text{ se } ps = \bot \end{cases}$$

Exemplo

$$\begin{array}{rcl} [\![\text{if } b \text{ then } S_1 \text{ else } S_2]\!]s & = & \text{cond}(\mathcal{B}[\![b]\!]s, [\![S_1, [\![S_2]\!]s]\!]s) \\ & = & \begin{cases} s' \text{ se } \mathcal{B}[\![b]\!] = \top \wedge [\![S_1]\!]s = s' \vee \\ \qquad \mathcal{B}[\![b]\!] = \bot \wedge [\![S_2]\!]s = s' \\ \oslash \text{ se } \mathcal{B}[\![b]\!] = \top \wedge [\![S_1]\!]s = \oslash \vee \\ \qquad \mathcal{B}[\![b]\!] = \bot \wedge [\![S_2]\!]s = \oslash \end{cases} \end{array}$$

Laço

Observações

- O efeito de `while b do S` deve ser igual ao efeito de `if b then (S; while b do S) else skip`
- Mas não pode-se usar

$$[\![\text{while } b \text{ do } S]\!] = \text{cond}(\mathcal{B}[\![b]\!], [\![\text{while } b \text{ do } S]\!] \circ [\![S]\!], id)$$

  porque $[\![\cdot]\!]$ não seria mais composicional
- Por isso, precisamos de usar um ponto fixo F tal que

$$F\ g = \text{cond}(\mathcal{B}[\![b]\!], g \circ [\![S]\!], id)$$

FIX

FIX : ((State → State) → (State → State)) → State → State

Exemplo

Trecho

`while` $\neg(x = 0)$ `do skip`

Ponto fixo

A funcional F' correspondente é:

$$(F'\ g)\ s = \begin{cases} g\ s \text{ se } s(x) \neq 0 \\ s \text{ se } x = 0 \end{cases}$$

Pontos fixos (?)

- $h\ s = \begin{cases} \oslash \text{ se } s(x) \neq 0 \\ s \text{ se } s(x) = 0 \end{cases}$
- $f\ s = \oslash$

Problemas

Funcionais com mais de um ponto fixo

F' tem mais de um ponto fixo porque toda função $g(x)$: State $\rightarrow$ State tal que $g\ s = s$ se $x = 0$ é ponto fixo de F'.

Funcionais sem ponto fixo

Seja $F_1\ g = \begin{cases} g_1 \text{ se } g = g_2 \\ g_2 \text{ senão} \end{cases}$

Se $g_1 \neq g_2$, não existe g_0 tal que, $F_1\ g_0 = g_0$.

Exercício

Assunto

- Determine a funcional F associado com

$$\text{while } \neg(x = 0) \text{ do } x := x - 1$$

  a partir da semântica denotacional dada.
- Determine quais das funções abaixo são pontos fixos de F
  - $g_1\ s = \emptyset$
  - $g_2\ s = \begin{cases} s[x \mapsto 0] \text{ se } s(x) \geq 0 \\ \emptyset \text{ se } x = 0 \end{cases}$
  - $g_3\ s = \begin{cases} s[x \mapsto 0] \text{ se } s(x) \geq 0 \\ s \text{ se } x = 0 \end{cases}$
  - $g_4\ s = s[x \mapsto 0$
  - $g_5\ s = s$

Exercício

Assunt

Considere o seguinte trecho de fatorial:

$$\text{while } \neg(x = 1) \text{ do } (y := y * x; x := x - 1)$$

Determine a funcional F associada a este trecho e pelo menos 2 pontos fixos diferentes.

Resumo

1. Extensões

2. Semântica denotacional

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