

Semânticas de Linguagens de Programação

Exercícios 5

Semântica para procedimentos e variáveis I

Semântica operacional natural para a linguagem Proc. Âmbitos de procedimentos e variáveis.

1. Considera o seguinte programa **c** em **Proc**:

```
begin var z := x;  
      x := y;  
      y := z  
end
```

Usando a semântica natural com âmbito dinâmico de **Proc** e considerando um ambiente env_P , constrói a árvore de derivação para execução de **c** com s_0 (i.e., $env_P \vdash \langle c, s_0 \rangle \longrightarrow s'$) tal que $s_0(x) = 3$ e $s_0(y) = 5$. Nota que neste caso env_P não vai ser usado pelo que podes omitir $env_P \vdash$ nas expressões deduzidas.

2. Considera o seguinte programa **f** em **Proc**:

```
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
```

Usando a semântica natural com âmbito dinâmico (para procedimentos e variáveis) constrói a árvore de derivação para execução com s_0 tal que $s_0(x) = 3$ e supondo que inicialmente temos um ambiente env_P .

3. Considera o seguinte programa **c** em **Proc** da pergunta 1. Usando a semântica natural com âmbito estático de **Proc** considera um ambiente env_P e um ambiente ev_0 onde $ev_0(x) = 1$, $ev_0(y) = 2$. Supõe ainda uma memória m_0 , com $m_0(1) = 3$, $m_0(2) = 5$ e $m_0(next) = 3$. Constrói a árvore de derivação para execução de **c** com

$$env_P, ev_0 \vdash \langle c, m_0 \rangle \longrightarrow m_1,$$

indicando o valor de m_1 .

4. Considera o programa **f** em **Proc** da pergunta 2.

Considera a semântica natural com âmbito estático para procedimentos e variáveis, sendo ep_0 um ambiente de procedimentos, $ev_0(x) = 1$, $ev_0(y) = 2$, $m_0(1) = 3$ e $m_0(next) = 3$. Calcula

$$ev_0, ep_0 \vdash \langle f, m_0 \rangle \longrightarrow m_1,$$

onde **c** é o comando (programa) dado.

5. Considera os seguintes comandos da linguagem **Proc**, c_0 :

```
begin var w := 0; var z := 0;
  proc inc is w := w + 1;
  while ¬ w = y do
    (z := z + x;
     call inc)
  end
```

e c_1 :

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

- (a) Seja $env_V(x) = 1$, $env_V(y) = 2$, $m_0(1) = 3$ e $m_0(2) = 2$, $m_0(next) = 3$ com $env_V : \mathbf{Var} \rightarrow \mathbf{Loc}$, $m_0 : \mathbf{Loc} \cup \{next\} \rightarrow \mathbb{Z}$ e $new(\ell) = \ell + 1$. Usando a semântica natural para a linguagem **Proc** com âmbito estático (para procedimentos e variáveis) determina m_1 tal que

$$env_V, env_P \vdash \langle c_0, m_0 \rangle \longrightarrow m_1$$

construindo a árvore de derivação.

- (b) Seja $s_0 : \mathbf{Var} \rightarrow \mathbb{Z}$ com $s_0(x) = 3$ e $s_0(y) = 2$. Usando a semântica natural para a linguagem **While** calcula $\langle c_1, s_0 \rangle \longrightarrow s_1$ indicando a árvore de derivação.
- (c) Qual é a relação entre s_1 e m_1 ?
- (d) Como poderias concluir que c_1 e c_0 são equivalentes para a semântica natural?
6. Repete os exercícios 1. da folha prática 3 usando a semântica natural de Proc com âmbito estático para variáveis. Divide o comando em subcomandos atribuindo um nome a cada um deles.
7. Modifica a sintaxe da declaração de procedimentos para aceitarem dois parâmetros *call-by-value* (passagem de parâmetros por valor).

$$D_P ::= \text{proc } p(x_1, x_2) \text{ is } c; D_P \mid$$

$$c ::= \dots \mid \text{call } p(a_1, a_2)$$

O ambiente dos procedimentos é agora

$$\mathbf{Env}_P = \mathbf{Pname} \hookrightarrow \mathbf{Com} \times \mathbf{Var} \times \mathbf{Var} \times \mathbf{Env}_V \times \mathbf{Env}_P$$

Modifica a semântica de **Proc** para estes constructores e em especial a semântica de **call**. Escreve um pequeno programa que ilustre o seu funcionamento.

8. Implementação em Haskell: estende a linguagem **While** a blocos e procedimentos (linguagem **Proc**), considerando também

```
module SNDProc
where
type Pname = String
type DV    = [ (Var, Aexp)]
type DP    = [ (Pname, Com)]
....
```

- (a) Implementa a semântica operacional natural com âmbito dinâmico para os procedimentos e variáveis $env_P \vdash \langle c, s \rangle \longrightarrow s'$.
- Define o sistema de transições $\rightarrow_D$ pela função $updV :: DV \rightarrow Mem \rightarrow Mem$
 - Define um ambiente de procedimentos `type EnvP = [ (String, Com) ]` e a função $udpP :: DP \rightarrow EnvP \rightarrow EnvP$. Nota: terás de definir analogos das funções `value` e `upd`.
 - Define uma função $snProcD :: EnvP \rightarrow Com \rightarrow Mem \rightarrow Mem$.
 - Testa com os exemplos dados nas aulas teóricas e práticas.
9. Implementação em Haskell: implementa a linguagem **Proc** considerando agora semântica operacional natural com âmbitos estáticos $env_V, env_P \vdash \langle c, mem \rangle \longrightarrow mem'$, seguindo as ideias acima e define uma função

`snProcE :: EnvV -> EnvP -> Com -> Mem -> Mem`

sendo que *Mem* deve corresponder às funções de $\mathbf{Loc} \cup \{next\} \rightarrow \mathbb{Z}$.

Semântica operacional natural para Proc com âmbitos dinâmicos

$$\begin{array}{l}
\text{att}_{sn} \quad env_P \vdash \langle x := a, s \rangle \longrightarrow s[x \mapsto \mathcal{A}[a] s] \\
\text{skip}_{sn} \quad env_P \vdash \langle \text{skip}, s \rangle \longrightarrow s \\
\text{comp}_{sn} \quad \frac{env_P \vdash \langle c_1, s \rangle \longrightarrow s', \quad env_P \vdash \langle c_2, s' \rangle \longrightarrow s''}{env_P \vdash \langle c_1; c_2, s \rangle \longrightarrow s''} \\
\text{if}_{sn}^v \quad \frac{env_P \vdash \langle c_1, s \rangle \longrightarrow s'}{env_P \vdash \langle \text{if } b \text{ then } c_1 \text{ else } c_2, s \rangle \longrightarrow s'} \\
\quad \text{se } \mathcal{B}[b] s = \text{true} \\
\text{if}_{sn}^f \quad \frac{env_P \vdash \langle c_2, s \rangle \longrightarrow s'}{env_P \vdash \langle \text{if } b \text{ then } c_1 \text{ else } c_2, s \rangle \longrightarrow s'} \\
\quad \text{se } \mathcal{B}[b] s = \text{false} \\
\text{while}_{sn}^v \quad \frac{env_P \vdash \langle c, s \rangle \longrightarrow s', env_P \vdash \langle \text{while } b \text{ do } c, s' \rangle \longrightarrow s''}{env_P \vdash \langle \text{while } b \text{ do } c, s \rangle \longrightarrow s''} \\
\quad \text{se } \mathcal{B}[b] s = \text{true} \\
\text{while}_{sn}^f \quad env_P \vdash \langle \text{while } b \text{ do } c, s \rangle \longrightarrow s \text{ se } \mathcal{B}[b] s = \text{false} \\
\text{block}_{sn} \quad \frac{\langle D_V, s \rangle \longrightarrow_D s', \quad upd_P(D_P, env_P) \vdash \langle c, s' \rangle \longrightarrow s''}{env_P \vdash \langle \text{begin } D_V \ D_P \ c \text{ end}, s \rangle \longrightarrow s''[DV(D_V) \mapsto s]} \\
\text{call}_{sn}^{rec} \quad \frac{env_P \vdash \langle c, s \rangle \longrightarrow s'}{env_P \vdash \langle \text{call } p, s \rangle \longrightarrow s'} \quad \text{onde } env_P(p) = c \\
\\
\text{var}_{sn} \quad \frac{\langle D_V, s[x \mapsto \mathcal{A}[a] s] \rangle \longrightarrow_D s'}{\langle \text{var } x := a; D_V, s \rangle \longrightarrow_D s'} \\
\text{empty}_{sn} \quad \langle \varepsilon, s \rangle \longrightarrow_D s
\end{array}$$

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

Semântica natural para Proc com âmbitos estáticos

Sendo $s = mem \circ env_V$, as regras $env_V, env_P \vdash \langle c, mem \rangle \longrightarrow mem'$ são:

$$\begin{array}{l}
\text{att}_{sn} \quad env_V, env_P \vdash \langle x := a, mem \rangle \longrightarrow mem[\ell \mapsto v] \\
\quad \text{onde } \ell = env_V(x), v = \mathcal{A}[[a]] s \\
\\
\text{skip}_{sn} \quad env_V, env_P \vdash \langle \text{skip}, mem \rangle \longrightarrow mem \\
\quad \quad \quad env_V, env_P \vdash \langle c_1, mem \rangle \longrightarrow mem' \\
\quad \quad \quad env_V, env_P \vdash \langle c_2, mem' \rangle \longrightarrow mem'' \\
\text{comp}_{sn} \quad \frac{env_V, env_P \vdash \langle c_1, mem \rangle \longrightarrow mem' \quad env_V, env_P \vdash \langle c_2, mem' \rangle \longrightarrow mem''}{env_V, env_P \vdash \langle c_1; c_2, mem \rangle \longrightarrow mem''} \\
\\
\text{if}^v_{sn} \quad \frac{env_V, env_P \vdash \langle c_1, mem \rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \text{if } b \text{ then } c_1 \text{ else } c_2, mem \rangle \longrightarrow mem'} \\
\quad \text{se } \mathcal{B}[[b]] s = \text{true} \\
\\
\text{if}^f_{sn} \quad \frac{env_V, env_P \vdash \langle c_2, mem \rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \text{if } b \text{ then } c_1 \text{ else } c_2, mem \rangle \longrightarrow mem'} \\
\quad \text{se } \mathcal{B}[[b]] s = \text{false} \\
\\
\text{while}^v_{sn} \quad \frac{env_V, env_P \vdash \langle c, mem \rangle \longrightarrow mem' \quad env_V, env_P \vdash \langle \text{while } b \text{ do } c, mem' \rangle \longrightarrow mem''}{env_V, env_P \vdash \langle \text{while } b \text{ do } c, mem \rangle \longrightarrow mem''} \\
\quad \text{se } \mathcal{B}[[b]] s = \text{true} \\
\\
\text{while}^f_{sn} \quad env_V, env_P \vdash \langle \text{while } b \text{ do } c, mem \rangle \longrightarrow mem \\
\quad \text{se } \mathcal{B}[[b]] s = \text{false} \\
\\
\text{block}_{sn} \quad \frac{\langle D_V, env_V, mem \rangle \longrightarrow_D (env'_V, mem') \quad env'_V, env'_P \vdash \langle c, mem' \rangle \longrightarrow mem''}{env_V, env_P \vdash \langle \text{begin } D_V \ D_P \ c \ \text{end}, mem \rangle \longrightarrow mem''} \\
\quad \text{onde } env'_P = upd_P(D_P, env'_V, env_P) \\
\\
\text{call}_{sn} \quad \frac{env'_V, env'_P \vdash \langle c, mem \rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \text{call } p, mem \rangle \longrightarrow mem'} \\
\quad \text{onde } env_P(p) = (c, env'_V, env'_P) \\
\\
\text{call}^{rec}_{sn} \quad \frac{env'_V, env'_P[p \mapsto (c, env'_V, env'_P)] \vdash \langle c, mem \rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \text{call } p, mem \rangle \longrightarrow mem'} \\
\quad \text{onde } env_P(p) = (c, env'_V, env'_P) \\
\\
\text{var}_{sn} \quad \frac{\langle D_V, env_V[x \mapsto \ell], mem[\ell \mapsto v][next \mapsto m] \rangle \longrightarrow_D (env'_V, mem')}{\langle \text{var } x := a; D_V, env_V, mem \rangle \longrightarrow_D (env'_V, mem')} \\
\quad \text{onde } v = \mathcal{A}[[a]] mem \circ env_V, \ell = mem(next), m = new(\ell) \\
\\
\text{empty}_{sn} \quad \langle \varepsilon, env_V, mem \rangle \longrightarrow_D (env_V, mem)
\end{array}$$

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