Semânticas de Linguagens de Programação
Exercícios 4
Semântica para procedimentos e variáveis

Conceitos de linguagens imperativas: âmbito de variáveis

1. Considera o seguinte programa em Pascal.

   program main;
   var a : *integer*;
   procedure $P1$
   begin
   writeln($'a =', a$);
   end;
   procedure $P2$
   var a : *integer*;
   begin
   $a := 10$;
   $P1$
   end;
   begin
   $a := 4$;
   $P2$
   end

   Supondo âmbito estático para a variáveis o que imprime o programa. E se o âmbito fosse dinâmico.

2. Escolhe uma linguagem de programação (com constructores imperativos) que te seja familiar. Quais as regras de âmbito de variáveis que usa? Exemplifica com um programa simples.

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

3. Considera o seguinte programa **c** 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
   ```

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

   (b) Usando a semântica natural com âmbito dinâmico para variáveis e estático para procedimentos supondo o mesmo valor para s_0.

(c) Usando a semântica natural com âmbito estático para procedimentos e variáveis. Considera que $ev_0(x) = 1, ev_0(y) = 2$, $m_0(1) = 3$, $m_0(next) = 3$ e calcula $ev_0, ep_0 \vdash \langle \mathbf{c}, m_0 \rangle \longrightarrow m_1$, onde **c** é o comando (programa) dado.

4. Considera os seguintes comandos da linguagem **Proc**, $\mathtt{c_0}$:

$$\begin{array}{l}\texttt{begin var } w \ := \ 0; \texttt{var } z \ := \ 0; \\ \quad \texttt{proc } inc \texttt{ is } w \ := \ w + 1; \\ \quad \texttt{while } \neg \, w \ = \ y \texttt{ do} \\ \quad\quad (z \ := \ z + x; \\ \quad\quad \texttt{call } inc) \\ \quad \texttt{end}\end{array}$$

e $\mathbf{c_1}$:

$$w := 0; z := 0; \ \texttt{while } \neg w = y \texttt{ do } (z := z + x; w := w + 1)$$

(a) Seja $env_V(x) = 1$, $env_V(y) = 2$, $m_0(1) = 3$ e $m_0(2) = 2$, $m_0(next) = 3$ com $env_V : \mathbf{Var} \to \mathbf{Loc}$, $m_0 : \mathbf{Loc} \cup \{next\} \to \mathbb{Z}$ 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

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

construindo a árvore de derivação.

(b) Seja $s_0 : \mathbf{Var} \to \mathbb{Z}$ com $s_0(x) = 3$ e $s_0(y) = 2$. Usando a semântica natural para a linguagem **While** calcula $\langle \mathbf{c_1}, s_0 \rangle \ \longrightarrow \ s_1$ indicando a árvore de derivação.

(c) Qual é a relação entre s_1 e m_1?

(d) Como poderias concluir que $\mathbf{c_1}$ e $\mathbf{c_0}$ são equivalentes para a semântica natural?

5. Repete os exercícios 1. e 2. 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.

6. Modifica a sintaxe da declaração de procedimentos para aceitarem dois parâmetros *call-by-value* (passagem de parâmetros por valor).

$$\begin{array}{rcl} D_P & ::= & \texttt{proc } p(x_1, x_2) \texttt{ is } c; \ D_P \mid \\ c & ::= & \ldots \mid \texttt{call } p(a_1, a_2) \end{array}$$

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.

7. 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 $env_P \vdash \langle \mathtt{c}, s\rangle \longrightarrow s'$.

i. Define o sistema de transições $\rightarrow_D$ pela função `updV :: DV -> Mem -> Mem`

ii. Define um ambiente de procedimentos `type EnvP = [ (String, Com) ]` e a função `udpP :: DP -> EnvP -> EnvP`. Nota: terás de definir analogos das funções `value` e `upd`.

iii. Define uma função `snProcD :: EnvP -> Com -> Mem -> Mem`.

iv. Testa com os exemplos dados nas aulas teóricas e práticas.

(b) Implementa a 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 âmbito dinâmico

$$\texttt{att}_{sn}\ env_P \vdash \langle x := a, s\rangle \longrightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$$

$$\texttt{skip}_{sn}\ env_P \vdash \langle \texttt{skip}, s\rangle \longrightarrow s$$

$$\texttt{comp}_{sn}\ \frac{env_P \vdash \langle c_1, s\rangle \longrightarrow s' \ ,\ env_P \vdash \langle c_2, s'\rangle \longrightarrow s''}{env_P \vdash \langle c_1; c_2, s\rangle \longrightarrow s''}$$

$$\texttt{if}^{v}{}_{sn}\ \frac{env_P \vdash \langle c_1, s\rangle \longrightarrow s'}{env_P \vdash \langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, s\rangle \longrightarrow s'} \quad \text{se } \mathcal{B}[\![b]\!]s = \mathsf{true}$$

$$\texttt{if}^{f}{}_{sn}\ \frac{env_P \vdash \langle c_2, s\rangle \longrightarrow s'}{env_P \vdash \langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, s\rangle \longrightarrow s'} \quad \text{se } \mathcal{B}[\![b]\!]s = \mathsf{false}$$

$$\texttt{while}^{v}{}_{sn}\ \frac{env_P \vdash \langle c, s\rangle \longrightarrow s', env_P \vdash \langle \texttt{while}\ b\ \texttt{do}\ c, s'\rangle \longrightarrow s''}{env_P \vdash \langle \texttt{while}\ b\ \texttt{do}\ c, s\rangle \longrightarrow s''} \quad \text{se } \mathcal{B}[\![b]\!]s = \mathsf{true}$$

$$\texttt{while}^{f}{}_{sn}\ env_P \vdash \langle \texttt{while}\ b\ \texttt{do}\ c, s\rangle \longrightarrow s \text{ se } \mathcal{B}[\![b]\!]s = \mathsf{false}$$

$$\texttt{block}_{sn}\ \frac{\langle D_V, s\rangle \longrightarrow_D s' \ ,\ upd_P(D_P, env_P) \vdash \langle c, s'\rangle \longrightarrow s''}{env_P \vdash \langle \texttt{begin}\ D_V\ D_P\ c\ \texttt{end}, s\rangle \longrightarrow s''[DV(D_V) \longmapsto s]}$$

$$\texttt{call}^{rec}{}_{sn}\ \frac{env_P \vdash \langle c, s\rangle \longrightarrow s'}{env_P \vdash \langle \texttt{call}\ p, s\rangle \longrightarrow s'} \quad \text{onde } env_P(p) = c$$

$$\texttt{var}_{sn}\quad \frac{\langle D_V, s[x \mapsto \mathcal{A}[\![a]\!]s]\rangle \longrightarrow_D s'}{\langle \texttt{var}\ x := a;\ D_V, s\rangle \longrightarrow_D s'}$$

$$\texttt{empty}_{sn}\quad \langle \varepsilon, s\rangle \longrightarrow_D s$$

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

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:

$$\texttt{att}_{sn}\ env_V, env_P \vdash \langle x := a, mem\rangle \longrightarrow mem[\ell \mapsto v]$$
onde $\ell = env_V(x), v = \mathcal{A}[\![a]\!]s$

$$\texttt{skip}_{sn}\ env_V, env_P \vdash \langle \texttt{skip}, mem\rangle \longrightarrow mem$$

$$\texttt{comp}_{sn}\ \frac{env_V, env_P \vdash \langle c_1, mem\rangle \longrightarrow mem' \quad env_V, env_P \vdash \langle c_2, mem'\rangle \longrightarrow mem''}{env_V, env_P \vdash \langle c_1; c_2, mem\rangle \longrightarrow mem''}$$

$$\texttt{if}^{v}{}_{sn}\ \frac{env_V, env_P \vdash \langle c_1, mem\rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, mem\rangle \longrightarrow mem'}$$
se $\mathcal{B}[\![b]\!]s = \mathsf{true}$

$$\texttt{if}^{f}{}_{sn}\ \frac{env_V, env_P \vdash \langle c_2, mem\rangle \longrightarrow mem'}{env_P \vdash \langle \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, mem\rangle \longrightarrow mem'}$$
se $\mathcal{B}[\![b]\!]s = \mathsf{false}$

$$\texttt{while}^{v}{}_{sn}\ \frac{env_V, env_P \vdash \langle c, mem\rangle \longrightarrow mem' \quad env_V, env_P \vdash \langle \texttt{while}\ b\ \texttt{do}\ c, mem'\rangle \longrightarrow mem''}{env_V, env_P \vdash \langle \texttt{while}\ b\ \texttt{do}\ c, mem\rangle \longrightarrow mem''}$$
se $\mathcal{B}[\![b]\!]s = \mathsf{true}$

$$\texttt{while}^{f}{}_{sn}\ env_V, env_P \vdash \langle \texttt{while}\ b\ \texttt{do}\ c, mem\rangle \longrightarrow mem$$
se $\mathcal{B}[\![b]\!]s = \mathsf{false}$

$$\texttt{block}_{sn}\ \frac{\langle D_V, env_V, mem\rangle \longrightarrow_D (env'_V, mem') \quad env'_V, env'_P \vdash \langle c, mem'\rangle \longrightarrow mem''}{env_V, env_P \vdash \langle \texttt{begin}\ D_V\ D_P\ c\ \texttt{end}, mem\rangle \longrightarrow mem''}$$
onde $env'_P = upd_P(D_P, env'_V, env_P)$

$$\texttt{call}_{sn}\ \frac{env'_V, env'_P \vdash \langle c, mem\rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \texttt{call}\ p, mem\rangle \longrightarrow mem'}$$
onde $env_P(p) = (c, env'_V, env'_P)$

$$\texttt{call}^{rec}{}_{sn}\ \frac{env'_V, env'_P[p \mapsto (c, env'_V, env'_P)] \vdash \langle c, mem\rangle \longrightarrow mem'}{env_V, env_P \vdash \langle \texttt{call}\ p, mem\rangle \longrightarrow mem'}$$
onde $env_P(p) = (c, env'_V, env'_P)$

$$\texttt{var}_{sn}\frac{\langle D_V, env_V[x \mapsto \ell], mem[\ell \mapsto v][next \mapsto m]\rangle \longrightarrow_D (env'_V, mem')}{\langle \texttt{var}\ x := a;\ D_V, env_V, mem\rangle \longrightarrow_D (env'_V, mem')}$$
onde $v = \mathcal{A}[\![a]\!]mem \circ env_V, \ell = mem(next), m = new(\ell)$

$$\texttt{empty}_{sn}\ \langle \varepsilon, env_V, mem\rangle \longrightarrow_D (env_V, mem)$$

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