Semânticas de Linguagens de Programação
Exercícios 4
Linguagens Imperativas. Semântica para procedimentos e variáveis I

Linguagens imperativas

1. `The Programming Languages Zoo`: `http://plzoo.andrej.com`
   (a) Instala a aplicação (instalando se necessário o OCAML e o mehnir). Para cada linguagem deves testar os exemplos e ver o `readme.md`
   (b) Calculadora: `calc`
      i. Escreve uma expressão regular para o léxico.
      ii. Escreve uma GIC para as expressões aritméticas
      iii. Desenha a árvore sintática para `4*5+2`.
      iv. A gramática é ambígua? Se for qual o problema para a implementação de linguagens? Como se pode resolver?
   (c) Calculadora com variáveis: `calc_var`: faz alguns testes e identifica diferenças com a anterior.
   (d) Para o paradigma imperativo/procedimental estuda a linguagem `comm`.
      i. Analisa a estrutura da linguagem indicando os seus construtores (ficheiro `sintax.ml` e relacionados).
      ii. Escreve uma GIC para as expressões aritméticas, booleanas e comandos.
      iii. Desenha uma árvore sintática para o programa: `new n:= 4*5+2 in print n`
      iv. Verifica no interpretador qual o valor calculado.
      v. Executa com a opção `--code` e vê qual o código produzido. Quais as instruções da máquina abstracta?
      vi. Escreve um programa para calcular o factorial de um inteiro.
      vii. Executa com a opção `--code` e vê qual o código produzido.

Conceitos de linguagens imperativas: âmbito de variáveis

2. 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.

3. 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.

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

$$\begin{array}{l}\texttt{begin var } z := x;\\ \qquad x := y;\\ \qquad y := z\\ \texttt{end}\end{array}$$

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., $env_P \vdash \langle \texttt{c}, s_0\rangle \longrightarrow s'$) 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.

5. 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
```

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 inicialmente temos um ambiente env_P.

6. 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.

Semântica operacional natural para Proc com âmbito dinâmico

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

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

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

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

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

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

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

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

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

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

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

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