Exercise Sheet 8: Semantics of Blocks and Procedures

Due date: June 23th. You can hand in your solutions at the start of the exercise class.

Exercise 1 (Understanding Local Scoping) 30%

Assume a variable y has already been declared in the outermost block. Give the value of variable y in the final state of each of the following programs in case of static as well as dynamic scoping.

a)[5%]

```
x := 3;
begin
  var x;
  x := 2;
  begin
    var x;
    x := 1;
    y := x;
  end;
end;
```

b)[5%]

```
x := 3;
begin
  var x;
  x := 2;
  y := x;
  begin
    var x;
    x := 1;
    y := x;
  end;
end
```

c)[5%]

```
x := 3;
begin
  var x;
  x := 2;
  y := x;
  begin
    var x; var y;
    x := 1;
    y := x;
  end;
end
```

d)[5%]

```
begin
  var x;
  proc P is y := x end;
  begin
    var x;
    x := 2;
    call P;
  end
end
```

e)[5%]

```
begin
  var x;
  proc P is x := 1 end;
  proc Q is call P end;
  begin
    var x;
    proc P is x := 2 end;
    x := 3;
    call Q;
    y := x;
  end
end
```

Exercise 2 (Blocks with Initialization of Local Variables) 20%

Assume we extend the WHILE programming language with blocks whose local variables are initialized (procedures are not considered in the extension).

$$\begin{aligned} v &::= \mathsf{var}\ x{:=}e;\ v \mid \epsilon \qquad (e \text{ ranges over } \mathsf{AExp}) \\ c &::= \ \dots \mid \mathsf{begin}\ v\ c\ \mathsf{end} \end{aligned}$$

(a) [10%] Modify the definition of the update function $\mathsf{upd}_v[\![\cdot]\!] : \mathsf{VDec} \times \mathsf{VEnv} \times \mathsf{Sto} \to \mathsf{VEnv} \times \mathsf{Sto}$ to account for the variable initialization:

$$\begin{aligned} \mathsf{upd}_v[\![\mathsf{var}\ x{:=}e;\ v]\!]\,(\rho,\sigma) &= \ \dots \\ \mathsf{upd}_v[\![\epsilon]\!]\,(\rho,\sigma) &= \ \dots \end{aligned}$$

(b) [10%] Now the execution relation depends only on the variable environment (there is no procedure environment). For instance, $\rho \vdash \langle c, \sigma\rangle \to \sigma'$ reads *"in (variable) environment ρ, statement c transforms store σ into store σ'"*. Complete the `(block)`–rule for defining the operational semantics of the language extension.

$$\frac{\dots}{\rho \vdash \langle \mathsf{begin}\ v\ c\ \mathsf{end}, \sigma\rangle \to \sigma''}$$

Exercise 3 (Procedures without Local Variables) 35%

In this exercise we consider a simpler model of procedures, where we asume no local variables. Procedures will manipulate the "global" program state and a call to a procedure will simple behave as unfolding its body. For the sake of concreteness we assume there are only procedures P_1 and P_2 of body $body_1$ and $body_2$ in Cmd. (We allow the possibility that these pair of procedures are mutually recursively defined, i.e. that $body_1$ and $body_2$ contain calls to P_1 and P_2.

The language syntax is extended by the following clause:

$$c \ ::= \ \ldots \mid \mathsf{call}\ P_1 \mid \mathsf{call}\ P_2 \ .$$

The semantics of the extension in defined in two steps. First we let $\mathsf{PInt} \triangleq (\Sigma \dashrightarrow \Sigma) \times (\Sigma \dashrightarrow \Sigma)$ be the set of procedure interpretation. We define semantic function

$$\mathfrak{C}[\![]\!]^* \colon \mathsf{Cmd} \to \mathsf{PInt} \to (\Sigma \dashrightarrow \Sigma)$$

which gives the denotation of a program w.r.t. a procedure interpretation. For any WHILE program c, $\mathfrak{C}[\![c]\!]^*_{(\theta_1,\theta_2)}$ is independent of procedure interpretation (θ_1, θ_2) and coincides with $\mathfrak{C}[\![c]\!]$. For procedure calls we simple extract its interpretation from the interpretation environment, i.e. $\mathfrak{C}[\![\mathsf{call}\ P_1]\!]^*_{(\theta_1,\theta_2)} = \theta_1$ and $\mathfrak{C}[\![\mathsf{call}\ P_2]\!]^*_{(\theta_1,\theta_2)} = \theta_2$.

(a) [20%] Determine the interpretation $(\theta_1^\star, \theta_2^\star)$ of procedures P_1 and P_2 induced by their bodies $body_1$ and $body_2$.
Hint: Have a look at the semantics of $\mathfrak{C}[\![body_1]\!]^*_{(\theta_1^\star,\theta_2^\star)}$ and $\mathfrak{C}[\![body_2]\!]^*_{(\theta_1^\star,\theta_2^\star)}$ first.

(b) [15%] Can the computation of $\theta_1^\star$ and $\theta_2^\star$ be simplified if we know that procedures P_1 and P_2 are not mutually recursive (but still recursive)?

Exercise 4 (Axiomatic Semantics with Local Variables) — 15%

(a) [5%] Let A be an assertion with free variables $\mathsf{var}(A)$. Define an assertion A' in which every $x \in \mathsf{var}(A)$ is replaced by a fresh existentially quantified variable x' such that $\models (A \Rightarrow A')$ holds.

(b) [10%] Recall the `WHILE` programming language extended with blocks whose local variables are initialized as introduced in Exercise 2. Extend the rules of axiomatic semantics to capture the local variable declarations and block definitions. You may assume that a sequence v of variable declarations contains no duplicates. For convenience, you may use $\mathsf{var}(v)$ ($\mathsf{var}(A)$) to denote the set of variables occuring in v (A) and $Exp(v)$ to denote the corresponding arithmetic expressions.