Series 5

Natural Operational Semantics of Languages **Block** and **Proc**

In this exercise series, we consider language **Block** and **Proc** and its natural operational semantics.

Natural Operational Semantics of Language Block

Exercise 47 (Computing the semantics of a program in Block) Compute the semantics of the following program on $\sigma_0 = [x \mapsto 0]$:

```
begin   var y := 1;
        (x := 1;
        begin var x := 2; y := x + 1 end
        x := y + x)
end
```

Natural Operational Semantics of Language Proc

Exercise 48 (Completing the semantics of Proc with static links) Complete the semantics of **Proc** with blocks and procedures with static links for variables and procedures.

Exercise 49 (Computing the semantics of a program) Let us consider the following code snippet seen in the lecture course:

```
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
```

Assume that variables x and y have been previously assigned, before entering this code, to x_0 and y_0 respectively. Moreover, assume that there is no procedure declared before entering this code snippet (i.e., the procedure environment is empty). Compute the semantics of this program according to the following three variants:

1. Dynamic links for procedures and variables.
2. Static links for procedures and dynamic link for variables.
3. Static links for procedures and variables.

Exercise 50 (Computing the semantics of a program) Let us consider the following program seen in the lecture course:

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

Compute the semantics of this program according to the following variants.

1. With static links for procedures, dynamic links for variables, and the recursive call rule,
2. With static links for procedures, dynamic links for variables and the non-recursive call rule.

Exercise 51 (Procedures as variables) We add the following statement to language **While** with blocks and procedures:

$$\textbf{Stm} ::= p := S.$$

Give a semantics to this language. Your semantics should be conservative wrt. the semantics of language **While**.

Exercise 52 (Procedures with parameters) We modify the syntax of procedures to allow parameters:

$$\begin{array}{lcl} S & \in & \textbf{Stm} \\ S & ::= & x := a \mid \mathsf{skip} \mid S_1; S_2 \\ & & \mid \mathsf{if}\ b\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \mid \mathsf{while}\ b\ \ \mathsf{do}\ S\ \mathsf{od} \\ & & \mid \mathsf{begin}\ D_V\ D_P;\ S\ \mathsf{end} \mid \mathsf{call}\ p(a_1, a_2) \\ D_V & ::= & \mathsf{var}\ \ x := a;\ D_V \mid \epsilon \\ D_P & ::= & \mathsf{proc}\ p(x_1, x_2)\ \mathsf{is}\ S;\ D_P \mid \epsilon \end{array}$$

We are interested in the semantics with static link for procedures and dynamic link for variables.

1. Modify the semantics of procedure declaration and call in order to obtain a semantics with call-by-value.
2. Same question with call-by-reference.
3. Same question with call by result.
4. Same question with call by value-result.

Exercise 53 (Completing a program) We are interested in the semantics with static links for variables and procedures. We extend language **Proc** with a write command: *write* x prints out the value of

$\sigma \circ \rho(x)$, but variable environment and storage function are left unchanged.
We consider the following program:

```
begin
        var  x := 2;
        begin
                        var  y := 7;
                        begin
                                        var  x := 5;
                                        var  y := 0;
                                        call p;
                        end
        end
end
```

1. Place the instructions proc p is $x := x * y$; and $write\ x$ so that the value 14 appears on screen.
2. Justify your answer by computing the output using the rules.