Université Joseph Fourier **Grenoble INP**
Master MOSIG M1 **2010-11**

Written Exam - November 2010
(some elements of solutions)

These 3 exercises are independent, but all them are based on the "While" language (studied in the lecture course), and those syntax is recalled in Appendix A.

Exercise 1. Procedures

The operational semantics of non-recursive procedures (without) parameters for the "While" language is recalled in Appendix B. We consider the following program P :

```
begin
     var x:=1 ;
     var y:=3 ;
     proc p1 is x:=x+1 ;
     /* location l0 */
     proc p2 is
          begin
                 var x:=4 ;
                 proc p is y:= x ;
                 /* location l1 */
                 call p ;
                 call p1 ;
                /* location l2 */
          end ;
     begin
        var x:=2 ;
        proc p1 is x:=x+y ;
        proc p is y:=x+y ;
        /* location l3 */
        call p2
        /* location l4 */
     end
```

Note that to properly answer this question you should follow the **execution order** of the program, namely `l0`, `l3`, `l4`, `l1` and `l2`

Q1. *We assume a dynamic binding for the variables and procedures. Give the values of the state σ at each location* `l0` *to* `l4` *and of the procedure environment Env_P at locations* `l1` *and* `l3`.

At `l0` :

$$\sigma = \{x \mapsto 1, y \mapsto 3\}$$

At `l1` :

$$\begin{aligned} \sigma &= \{x \mapsto 4, y \mapsto 3\} \\ Env_P &= \{p1 \mapsto \texttt{x} := \texttt{x} + \texttt{y},\ \ p \mapsto \texttt{y} := \texttt{x}\ \ p2 \mapsto \texttt{begin...end}\} \end{aligned}$$

At `l2` :

$$\sigma = \{x \mapsto 8, y \mapsto 4\}$$

At `l3` :

$$\begin{aligned} \sigma &= \{x \mapsto 2, y \mapsto 3\} \\ Env_P &= \{p2 \mapsto \texttt{begin...end},\ \ p1 \mapsto \texttt{x} := \texttt{x} + \texttt{y},\ \ p \mapsto \texttt{y} := \texttt{x} + \texttt{y}\} \end{aligned}$$

At `l4` :

$$\sigma = \{x \mapsto 2, y \mapsto 4\}$$

Q2. *Same question when we assume a dynamic binding for the variables and a static binding for the procedures.*

At `l0` :

$$\sigma = \{x \mapsto 1, y \mapsto 3\}$$

At `l1` :

$$\begin{aligned} \sigma &= \{x \mapsto 4, y \mapsto 3\} \\ Env_P^1 &= \{p1 \mapsto (\texttt{x} := \texttt{x} + \texttt{1}, \emptyset)\} \ \text{ before declaration of p} \\ Env_P &= Env_P^1[\{p \mapsto (\texttt{y} := \texttt{x}, Env_P^1)\} \ \textbf{the answer} \end{aligned}$$

At `l2` :

$$\sigma = \{x \mapsto 5, y \mapsto 4\}$$

At `l3` :

$$\begin{aligned} \sigma &= \{x \mapsto 2, y \mapsto 3\} \\ Env_P^1 &= \{p1 \mapsto (\texttt{x} := \texttt{x} + \texttt{1}, \emptyset)\} \ \text{ external declaration of p1} \\ Env_P^2 &= Env_P^1[\{p2 \mapsto (\texttt{begin...end}, Env_P^1)] \ \text{ external declaration of p2} \\ Env_P^3 &= Env_P^2[\{p1 \mapsto (\texttt{x} := \texttt{x} + \texttt{y}, Env_P^2)] \ \text{ re-declaration of p1} \\ Env_P &= Env_P^3[\{p \mapsto (\texttt{y} := \texttt{x} + \texttt{y}, Env_P^3)] \ \textbf{the answer} \end{aligned}$$

At `l4` :

$$\sigma = \{x \mapsto 2, y \mapsto 4\}$$

Exercise 1. An extension of the "While" language

We extend the "While" language by introducing a new statement whose syntax is the following :

$\texttt{for}\ x := a_1\ \texttt{to}\ a_2\ \texttt{do}\ S$

Q1. *Give a "While" program $P2$ such that $P2$ is equivalent to $P1$ and $P2$ does not contain any "for" statement.*

```
var x := 0 ;
begin
    var v1:=1 ; var v2:=100 ;
    var x:=v1 ;
    while x <= v2
          print (x) ;
          x := x + 1 ;
end ;
-- useless to restore x to its previous value,
-- since we used a nested block with a local x inside
print(x)
```

Q2. *Give a semantic rules for the "for" statement :*

1. *Using the "while" statement*
2. *Without using the "while" statement.*

1. A solution using the `while` statement :
   - If $\mathcal{A}[a_1]\sigma \leq \mathcal{A}[a_2]\sigma$ then

$$\frac{(S, \sigma[x \mapsto v_1]) \to \sigma', \quad (\mathsf{while}\ x < a_2\ \ \mathsf{do}\ S\ , \sigma') \to \sigma''}{(\texttt{for}\ \ x := a_1\ \texttt{to}\ \ a_2\ \texttt{do}\ S, \sigma) \to \sigma''[x \mapsto \sigma(x)]}$$

where $v_1 = \mathcal{A}[a_1]\sigma$ and $v_2 = \mathcal{A}[a_2]\sigma$

   - If $\mathcal{A}[a_1]\sigma > \mathcal{A}[a_2]\sigma$ then

$$(\texttt{for}\ \ x := a_1\ \texttt{to}\ \ a_2\ \texttt{do}\ S, \sigma) \to \sigma$$

2. A solution without using the `while` statement :
   - If $\mathcal{A}[a_1]\sigma \leq \mathcal{A}[a_2]\sigma$ then

$$\frac{(S, \sigma[x \mapsto v_1]) \to \sigma', \quad (\texttt{for}\ \ x := \sigma'(x)\ \texttt{to}\ \ a_2\ \texttt{do}\ S, \sigma') \to \sigma''}{(\texttt{for}\ \ x := a_1\ \texttt{to}\ \ a_2\ \texttt{do}\ S, \sigma) \to \sigma''[x \mapsto \sigma(x)]}$$

where $v_1 = \mathcal{A}[a_1]\sigma$ and $v_2 = \mathcal{A}[a_2]\sigma$

   - If $\mathcal{A}[a_1]\sigma > \mathcal{A}[a_2]\sigma$ then

$$(\texttt{for}\ \ x := a_1\ \texttt{to}\ \ a_2\ \texttt{do}\ S, \sigma) \to \sigma$$

Q3. *According to your rules, what are the values printed when executing the following program $P3$:*

The values printed are `0, 8, 9, 0`.

Exercise 3. Code generation for the AM machine - 6 pts.

Q1. *Extend the function $\mathcal{CS}$ to deal with the new "While" statement* `repeat` S `until` b

$$\mathcal{CS}[\texttt{repeat } S \texttt{ until } b] = \mathcal{CS}[S] : \texttt{LOOP}(\mathcal{CB}[\neg b], \mathcal{CS}[S])$$

Q2. *Give the code produced for the following statement :*

```
repeat
     x := x+2 ;
     y := y+1 ;
until x=y ;
```

```
Push-2:Fetch(x):Add:Store(x):Push-1:Fetch(y):Add:Store(y):
Loop(
      Fetch(y):Fetch(x):EQ:Neg,
      Push-2:Fetch(x):Add:Store(x):Push-1:Fetch(y):Add:Store(y):
)
```