UFR-IMAG
Université Joseph Fourier

Programming Language and Compiler Design,
2010/2011
Marion Daubignard
Yassine Lakhnech
Laurent Mounier

Homework - Version A.

Exercise 1

We define the syntactic category of letters $L = \{a, b, c\}$. By l we denote a meta-variable ranging over L. We define inductively a set of words W by the following BNF:

$$w := l \mid w_1 c w_2 b \mid cwab$$

where w, w_1, w_2 are meta-variables ranging over the set of words W. Of the following two statements, one is wrong and one is right. You get to tell which is which and justify your answers by either a proof (by induction) or a counter-example.

1. The length of words in W is congruent to 1 modulo 3.
2. All words in W finish with a letter b.

The first statement is true. We provide a proof by structural induction.
Basic case:
The atoms of the set are letters. Every letter $l \in L$ has length 1, which is indeed congruent to 1 mod 3.
Case of rule 'if $w_1, w_2 \in W$ then $w_1 c w_2 b \in W$':
our induction hypothesis is that w_1 and w_2 verify the property we want to prove, i.e. their respective length is congruent to 1 mod 3. The length of the word $w_1 c w_2 b$ is 2 plus that of w_1 and that of w_2. Hence we have a global length congruent to (2+1+1) mod 3= 1 mod 3.
Case of rule 'if $w \in W$ then $cwab \in W$':
our induction hypothesis is that w verifies the property we want to prove, i.e. its length is congruent to 1 mod 3. Hence the length of $cwab$ is congruent to (1+3) mod 3= 1 mod 3.
In conclusion, we have proven by structural induction that for every word $w \in W$, the length of w is congruent to 1 mod 3.

The second statement is wrong: a is a letter, hence by the first rule it belongs to W, and word a does not finish with b.

Exercise 2

We consider the following program.

```
begin  var y := 2;
       var x := 71 − y;
       proc plop is x := y + x;
       begin var y := 1;
       proc p is call plop;
       proc plop is x := y;
       call p;
       end
       call plop;
end
```

You have been presented three different semantics for the While language with blocks and procedures: one with dynamic links for variables and procedures, another with dynamic links for variables but static links for procedures, and finally one with static links for variables and procedures.
What values are associated to x and y at the end of this program according to each of the three semantics you know ? Justify your answer (you can either draw the tree or precise the state or the variable environment and the storage function after each ';').

$$
\begin{array}{ll}
\textsf{begin} & D_V = \left\{ \begin{array}{l} \textsf{var } \ y := 2; \\ \textsf{var } \ x := 71 - y; \end{array} \right. \\
& D_P = \left\{ \ \textsf{proc } plop \textsf{ is } x := y + x; \right. \\
& S = \left\{ \begin{array}{l}
S_1 = \left\{ \begin{array}{ll}
\textsf{begin} & D'_V = \left\{ \ \textsf{var } \ y := 1; \right. \\
& D'_P = \left\{ \begin{array}{l} \textsf{proc } p \textsf{ is call } plop; \\ \textsf{proc } plop \textsf{ is } x := y; \end{array} \right. \\
& S' = \left\{ \ \textsf{call } p; \right. \\
\textsf{end} &
\end{array} \right. \\
S_2 = \left\{ \ \textsf{call } plop; \right.
\end{array} \right. \\
\textsf{end} &
\end{array}
$$

$$
\dfrac{< D_V, \sigma > \xrightarrow{V} \sigma_2 \qquad \dfrac{\dfrac{< D'_V, \sigma_2 > \xrightarrow{V} \sigma_6 \qquad \dfrac{\dfrac{\dfrac{}{(env^3_P, x := y, \sigma_6) \rightarrow \sigma_5}\textsc{Asg}}{(env^3_P, call\ plop, \sigma_6) \rightarrow \sigma_5}\textsc{Call}}{(env^3_P, S', \sigma_6) \rightarrow \sigma_5}\textsc{Call}}{(env^1_P, S_1, \sigma_2) \rightarrow \sigma_4}\textsc{Bl} \qquad \dfrac{(env^1_P, x := y + x, \sigma_4) \rightarrow \sigma_3\ \textsc{Asg}}{(env^1_P, S_2, \sigma_4) \rightarrow \sigma_3}\textsc{Call}}{(env^1_P, S, \sigma_2) \rightarrow \sigma_3}\textsc{Seq}}{(env_P, \textsf{begin}\ D_V\ D_P\ S, \sigma) \rightarrow \sigma_1}\textsc{Bl}
$$

where : σ and env_P are initial state and environment

$\sigma_2 = \sigma[x \mapsto 69, y \mapsto 2]$

$\sigma_6 = \sigma_2[y \mapsto 1](= \sigma[x \mapsto 69, y \mapsto 1])$

$env^3_P = env^2_P[plop \mapsto (x := y)](= env_P[p \mapsto (call\ plop), plop \mapsto (x := y)])$

$\sigma_5 = \sigma_6[x \mapsto 1] = \sigma[x \mapsto 1, y \mapsto 1]$

$\sigma_4 = \sigma_5[DV(D'_V) \mapsto \sigma_2], DV(D'_V) = \{y\}$

$\sigma_3 = \sigma_4[x \mapsto 3] = \sigma[x \mapsto 3, y \mapsto 2]$

$env^1_P = env_P[plop \mapsto (x := y + x)]$

$env^2_P = env^1_P[p \mapsto (call\ plop)]$

so $\sigma_4 = \sigma_5[y \mapsto 2] = \sigma[x \mapsto 1, y \mapsto 2]$

$\sigma_1 = \sigma_3[DV(D_V) \mapsto \sigma], DV(D_V) = \{x, y\},$ so $\sigma_1 = \sigma$

Figure 1: Dynamic-dynamic semantics

$$
\textsc{Bl}\ \dfrac{< D_V, \sigma > \stackrel{V}{\longrightarrow} \sigma_2 \qquad \dfrac{\textsc{Bl}\ \dfrac{< D'_V, \sigma_2 > \stackrel{V}{\longrightarrow} \sigma_6 \qquad \dfrac{\dfrac{\dfrac{}{(env_P, x := y + x, \sigma_6) \to \sigma_5}\ \textsc{Asg}}{(env^1_P, call\ plop, \sigma_6) \to \sigma_5}\ \textsc{Call}}{(env^3_P, S', \sigma_6) \to \sigma_5}\ \textsc{Call}}{(env^1_P, S_1, \sigma_2) \to \sigma_4} \qquad \dfrac{(env_P, x := y + x, \sigma_4) \to \sigma_3\ \textsc{Asg}}{(env^1_P, S_2, \sigma_4) \to \sigma_3}\ \textsc{Call}}{(env^1_P, S, \sigma_2) \to \sigma_3}}{(env_P, \mathsf{begin}\ D_V\ D_P\ S, \sigma) \to \sigma_1}
$$

where :

σ and env_P are initial state and environment

$\sigma_2 = \sigma[x \mapsto 69, y \mapsto 2]$

$env^1_P = env_P[plop \mapsto (y := x + y, env_P)]$

$\sigma_6 = \sigma_2[y \mapsto 1](= \sigma[x \mapsto 69, y \mapsto 1])$

There are two steps to compute the proc. env. update:

$env^2_P = env^1_P[p \mapsto (call\ plop, env^1_P)]$

$env^3_P = env^2_P[plop \mapsto (x := y, env^2_P)]$

Note that $env^3_P = env_P[p \mapsto (call\ plop, env^1_P), plop \mapsto (x := y, env^2_P)]$

$\sigma_5 = \sigma_6[x \mapsto 70] = \sigma[x \mapsto 70, y \mapsto 1]$

$\sigma_4 = \sigma_5[DV(D'_V) \mapsto \sigma_2], DV(D'_V) = \{y\},$ so $\sigma_4 = \sigma_5[y \mapsto 2] = \sigma[x \mapsto 70, y \mapsto 2]$

$\sigma_3 = \sigma_4[x \mapsto 72] = \sigma[x \mapsto 72, y \mapsto 2]$

$\sigma_1 = \sigma$

Figure 2: Static-dynamic semantics

$$\textsc{Bl}\ \frac{< D_V, \sigma > \xrightarrow{V} (env_V^2, sto^2) \qquad \dfrac{T_1 \qquad T_2}{(env_P^1, S, env_V^2, sto^2) \rightarrow (env_V^3, sto^3)}\ \textsc{Seq}}{(env_P, \mathsf{begin}\ D_V\ D_P\ S, env_V, sto) \rightarrow (env_V^1, sto^1)}$$

with T_1 :

$$\textsc{Bl}\ \frac{< D'_V, env_V^2, sto^2 > \xrightarrow{V} (env_V^6, sto^6) \qquad \dfrac{\dfrac{\dfrac{}{(env_P, x := y + x, env_V^2, sto^6) \rightarrow (env_V^2, sto^5)}\ \textsc{Asg}}{(env_P^1, call\ plop, env_V^6, sto^6) \rightarrow (env_V^5, sto^5)}\ \textsc{Call}}{(env_P^3, S', env_V^6, sto^6) \rightarrow (env_V^5, sto^5)}\ \textsc{Call}}{(env_P^1, S_1, env_V^2, sto^2) \rightarrow (env_V^4, sto^4)}$$

with T_2 being :

$$\frac{\dfrac{}{(env_P^1, x := y + x, env_V^4, sto^4) \rightarrow (env_V^3, sto^3)}\ \textsc{Asg}}{(env_P^1, S_2, env_V^4, sto^4) \rightarrow (env_V^3, sto^3)}\ \textsc{Call}$$

where :

σ and env_P are initial state and environment

$env_V^2 = env_V[y \mapsto 1, x \mapsto 2], sto^2 = sto[1 \mapsto 2, 2 \mapsto 69]$ $\qquad env_P^1 = env_P[plop \mapsto (y := x + y, env_P, env_V^2)]$

$env_V^6 = env_V[y \mapsto 3, x \mapsto 2], sto^6 = sto[1 \mapsto 2, 2 \mapsto 69, 3 \mapsto 1]$

There are two steps to compute the proc. env. update:

$env_P^2 = env_P^1[p \mapsto (call\ plop, env_P^1, env_V^6)]$ $\qquad env_P^3 = env_P^2[plop \mapsto (x := y, env_P^2, env_V^6)]$

Note that $env_P^3 = env_P[p \mapsto (call\ plop, env_P^1, env_V^6), plop \mapsto (x := y, env_P^2, env_V^6)]$

$env_V^5 = env_V^6, sto^5 = sto[1 \mapsto 2, 2 \mapsto 71, 3 \mapsto 1]$

$env_V^4 = env_V^2 = env_V[y \mapsto 1, x \mapsto 2], sto^4 = sto[1 \mapsto 2, 2 \mapsto 71]$

$env_V^3 = env_V^4 = env_V[y \mapsto 1, x \mapsto 2], sto^3 = sto[1 \mapsto 2, 2 \mapsto 73]$

$env_V^1 = env_V, sto^1 = sto.$

Figure 3: Static-static semantics