Univ. Grenoble Alpes **UFR IM2AG**
MOSIG 1 **Academic year 2016-2017**

PLCD

Mid-term Exam, November 9, 2016

Guidelines and information:

- Duration : 1 hour 30 minutes
- All documents from the course and the tutorial are authorized.
- Exercises are **independent**.
- The grading scale is indicative.

Part I: Operational Semantics (10 points)

We consider the syntax of arithmetical expressions seen in the course and extend it with two new operators: one operator for *pre-incrementation* and one operator for *post-incrementation*. The new syntax of arithmetical expressions is the following:

$$a \quad ::= \quad n \mid x \mid a_1 + a_2 \mid + + x \mid x + +$$

We note **Aexp'** the set of the arithmetical expressions built with this new syntax. Informally, the value of expression $x + +$ is the value of variable x, then we associate to x its former value augmented by 1. Similarly, the value of expression $+ + x$ is obtained by augmenting by 1 the value associated to x, and then returning the new value. For instance, in a state where x is equal to 3, $x + +$ is equal to 3 and the new value of x is 4 whereas $+ + x$ is equal to 4 and the new value of x is 4.

Notice that the evaluation of expression can modify the state.

Configurations for arithmetical expressions are thus given by $(\textbf{Aexp'} \times \textbf{State}) \cup (\mathbb{Z} \times \textbf{State})$ (we encode the value of the expression, and the state obtained by side-effect).

Rules are of the following form: $(a, \sigma) \longrightarrow (v, \sigma')$.

Do not use function $\mathcal{A}$ seen in the course.

Answer of exercise 1

1. A constant (resp. variable) does not modify the value of a state, and its value is $\mathcal{N}(n)$ (resp. $\sigma(x)$). Therefore, we deduce the following axioms :

$$(n, \sigma) \to (\mathcal{N}(n), \sigma) \text{ and } (x, \sigma) \to (\sigma(x), \sigma)$$

For the addition, we consider the possible side effects of the left-hand member to evaluate the right member:

$$\frac{(a_1, \sigma) \to (v_1, \sigma_1) \quad (a_2, \sigma_1) \to (v_2, \sigma_2)}{(a_1 + a_2, \sigma) \to (v_1 + v_2, \sigma_2)}$$

For expressions of the forms $++x$ and $x++$, we modify the state according to the informal semantics:

$$(++x, \sigma) \to (\sigma(x) + 1, \sigma[x \mapsto \sigma(x) + 1])$$
$$(x++, \sigma) \to (\sigma(x), \sigma[x \mapsto \sigma(x) + 1])$$

2. Applying the rules on the provided expression, we find: $(x + (x++), x \mapsto 3) \to (6, x \mapsto 4)$, which is as expected.

3. The rule is immediate. Make sure to consider the possible side-effects of expression a.

$$\frac{(a, \sigma) \to (v, \sigma')}{(x{+}{=}\ a, \sigma) \ \to \sigma'[x \mapsto v + \sigma(x)]}$$

4. Applying the rules on the provided expression, we find: $(x{+}{=}\ (++y), [x \mapsto 3, y \mapsto 5]) \Rightarrow [y \mapsto 6, x \mapsto 9]$, which is as expected.

5.
   - The statements are semantically equivalent. Let us suppose that $(a, \sigma) \longrightarrow (v, \sigma')$, we have:

     $$\cfrac{\cfrac{(x, \sigma) \to (\sigma(x), \sigma) \quad (a, \sigma) \to (v, \sigma')}{(x + a, \sigma) \to (\sigma(x) + v, \sigma')}}{(x := x + a, \sigma) \to \sigma'[x \mapsto v + \sigma(x)]}$$

     The final state is as the one obtained by executing $x+ = a$ in σ.
   - The statements are not semantically equivalent. Indeed, we have:

     $$\cfrac{\cfrac{(a, \sigma) \to (v, \sigma') \quad (x, \sigma') \to (\sigma'(x), \sigma')}{(a + x, \sigma) \to (\sigma'(x) + v, \sigma')}}{(x := a + x, \sigma) \to \sigma'[x \mapsto v + \sigma'(x)]}$$

     One can also provide a counter-example. Take $\sigma = [x \mapsto 1]$ and $a = ++x$. We have: $(x+ := a, \sigma) \to [x \mapsto 3]$ and $(x := a + x, \sigma) \to [x \mapsto 4]$.

Part II: Types (5 points)

In this exercise, we are interested in expressions of language `While` seen in the course. The idea is to consider the sign rule as a type system. This rule states for instance that the sum of two positive integers is a positive integer, that the product of two negative integers is positive, etc. However, the difference of two positive integers is an integer. We first recall the (slightly modified) syntax seen in the course:

$$\begin{array}{lcl} a & := & p \mid n \mid x \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2 \\ b & := & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \end{array}$$

We have thus two syntactic categories *Aexp* and *Bexp*, respectively for arithmetical and boolean expressions.

The modification consists in replacing the denotation of an integer by a denotation of a positive or null integer, noted p and a denotation of a strictly negative integer, noted n. We define 3 types `Pos, Neg, Int` that designate respectively positive or null integer, strictly negative integers, and (relative) integers. We have a natural order on types: $\texttt{Pos} < \texttt{Int}$ and $\texttt{Neg} < \texttt{Int}$.

We consider the type environment that associates a type to each variable:

$$\texttt{Env} = \texttt{Var} \longrightarrow \{\texttt{Pos}, \texttt{Neg}, \texttt{Int}\}.$$

Configurations are given by:

- $\Gamma \vdash a : t$ with $\Gamma \in \mathit{Env}$, $a \in \mathit{Aexp}$ and $t \in \{\texttt{Pos}, \texttt{Neg}, \texttt{Int}\}$ for arithmetical expressions,
- $\Gamma \vdash b : t$ with $\Gamma \in \mathit{Env}$, $b \in \mathit{Bexp}$ and $t \in \{\texttt{Bool}\}$ for boolean expressions.

We note Γ an element of the environment and $\tau, \tau_1, \ldots$ a type.

Part A: Arithmetical Expressions

We give below the rules for constants, variables, and some of the rules for addition.

$$\Gamma \vdash p : \texttt{Pos} \qquad \Gamma \vdash n : \texttt{Neg} \qquad \Gamma \vdash x : \Gamma(x)$$

$$\frac{\Gamma \vdash a_1 : \texttt{Pos} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 + a_2 : \texttt{Pos}} \qquad \frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Neg}}{\Gamma \vdash a_1 + a_2 : \texttt{Neg}}$$

$$\frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 + a_2 : \texttt{Int}}$$

Notice that the addition of two positive (resp. negative) integers results in a positive (resp. negative) integer.

However, the addition of a positive and a negative integer is an integer (this reflects the loss of information).

Answer of exercise 2

1. For subtraction:

$$\frac{\Gamma \vdash a_1 : \texttt{Pos} \quad \Gamma \vdash a_2 : \texttt{Neg}}{\Gamma \vdash a_1 - a_2 : \texttt{Pos}} \qquad \frac{\Gamma \vdash a_1 : \texttt{Pos} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 - a_2 : \texttt{Int}}$$

$$\frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 - a_2 : \texttt{Neg}} \qquad \frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Neg}}{\Gamma \vdash a_1 - a_2 : \texttt{Int}}$$

$$\frac{\Gamma \vdash a_1 : \texttt{Int} \quad \Gamma \vdash a_2 : x \in \{\texttt{Int}, \texttt{Neg}, \texttt{Pos}\}}{\Gamma \vdash a_1 - a_2 : \texttt{Int}} \quad \frac{\Gamma \vdash a_1 : x \in \{\texttt{Int}, \texttt{Neg}, \texttt{Pos}\} \quad \Gamma \vdash a_2 : \texttt{Int}}{\Gamma \vdash a_1 - a_2 : \texttt{Int}}$$

2. For multiplication:

$$\frac{\Gamma \vdash a_1 : \texttt{Pos} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 * a_2 : \texttt{Pos}} \qquad \frac{\Gamma \vdash a_1 : \texttt{Pos} \quad \Gamma \vdash a_2 : \texttt{Neg}}{\Gamma \vdash a_1 * a_2 : \texttt{Int}}$$

$$\frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 * a_2 : \texttt{Int}} \qquad \frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Neg}}{\Gamma \vdash a_1 * a_2 : \texttt{Pos}}$$

$$\frac{\Gamma \vdash a_1 : x \in \{\texttt{Int}, \texttt{Neg}, \texttt{Pos}\} \quad \Gamma \vdash a_2 : \texttt{Int}}{\Gamma \vdash a_1 * a_2 : \texttt{Int}} \quad \frac{\Gamma \vdash a_1 : \texttt{Int} \quad \Gamma \vdash a_2 : x \in \{\texttt{Int}, \texttt{Neg}, \texttt{Pos}\}}{\Gamma \vdash a_1 * a_2 : \texttt{Pos}}$$

Part B: Boolean Expressions

For boolean expressions, we consider the following syntax:

$$b \quad ::= \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2$$

Answer of exercise 3

1. We propose two solutions.

   (a) A solution not using types `True` and `False`.
   There are several solutions depending on how strict one wants the type system to be. The following options are not exhaustive. We start with the strictest option and go to the least strict one, in order.

   – Option 1: We allow the comparison between expressions of the same type:

$$\frac{\Gamma \vdash a_1 : x \quad \Gamma \vdash a_2 : x \quad x \in \{\texttt{Neg}, \texttt{Pos}, \texttt{Int}\}}{\Gamma \vdash a_1 = a_2 : \texttt{Bool}}$$

   – Option 2: We allow the comparison of expressions of types that are comparable according to the (partial) order defined between types:

$$\frac{\Gamma \vdash a_1 : x \quad \Gamma \vdash a_2 : y \quad x, y \in \{\texttt{Neg}, \texttt{Pos}, \texttt{Int}\} \quad x < y \vee y < x \vee x = y}{\Gamma \vdash a_1 = a_2 : \texttt{Bool}}$$

– Option 3: We allow the comparison between expressions of any type:

$$\frac{\Gamma \vdash a_1 : x \quad \Gamma \vdash a_2 : y \quad x, y \in \{\texttt{Neg}, \texttt{Pos}, \texttt{Int}\}}{\Gamma \vdash a_1 = a_2 : \texttt{Bool}}$$

(b) A solution using types `True` and `False`. In that case, we must allow for comparing values of different types. We provide rules given in their precedence order.

$$\frac{\Gamma \vdash a_1 : x \quad \Gamma \vdash a_2 : y \quad (x = \texttt{Neg} \wedge y = \texttt{Pos}) \vee (y = \texttt{Neg} \wedge x = \texttt{Pos})}{\Gamma \vdash a_1 = a_2 : \texttt{False}}$$

$$\frac{\Gamma \vdash a_1 : x \quad \Gamma \vdash a_2 : y \quad x, y \in \{\texttt{Neg}, \texttt{Pos}, \texttt{Int}\}}{\Gamma \vdash a_1 = a_2 : \texttt{Bool}}$$

2. We provide rules given in their precedence order.

$$\frac{\Gamma \vdash a_1 : \texttt{Neg} \quad \Gamma \vdash a_2 : \texttt{Pos}}{\Gamma \vdash a_1 \leq a_2 : \texttt{True}} \qquad \frac{\Gamma \vdash a_2 : \texttt{Neg} \quad \Gamma \vdash a_1 : \texttt{Pos}}{\Gamma \vdash a_1 \leq a_2 : \texttt{False}}$$

$$\frac{\Gamma \vdash a_1 : x \quad \Gamma \vdash a_2 : y \quad x, y \in \{\texttt{Neg}, \texttt{Pos}, \texttt{Int}\}}{\Gamma \vdash a_1 \leq a_2 : \texttt{Bool}}$$

Part III: Optimization (5 points)

We consider the control-flow graph in figure 1 :

Answer of exercise 4

1. Let $e_1 = i + j, e_2 = i - j$ and $\Sigma = \{e_1, e_2\}$. We have:

B	$Gen(b)$	$Kill(b)$
B_1	e_1	e_1, e_2
B_2	$\emptyset$	$\emptyset$
B_3	e_1, e_2	$\emptyset$
B_4	e_1	$\emptyset$
B_5	e_1, e_2	$\emptyset$
B_6	e_2	$\emptyset$

2.

B	$pre(b)$	$In(b)$	$Out(b)$	$In(b)$	$Out(b)$	$In(b)$	$Out(b)$	$In(b)$	$Out(b)$	$In(b)$	$Out(b)$
B_1	$\emptyset$	$\emptyset$	e_1	$\emptyset$	e_1	$\emptyset$	e_1	$\emptyset$	e_1	$\emptyset$	e_1
B_2	B_1, B_4, B_6	Σ	Σ	e_1	e_1	e_1	e_1	e_1	e_1	e_1	e_1
B_3	B_2	Σ	Σ	Σ	Σ	e_1	Σ	e_1	Σ	e_1	Σ
B_4	B_3	Σ	Σ	Σ	Σ	Σ	Σ	Σ	Σ	Σ	Σ
B_5	B_2	Σ	Σ	Σ	Σ	e_1	Σ	e_1	Σ	e_1	Σ
B_6	B_3	Σ	Σ	Σ	Σ	e_1	Σ	Σ	Σ	Σ	Σ

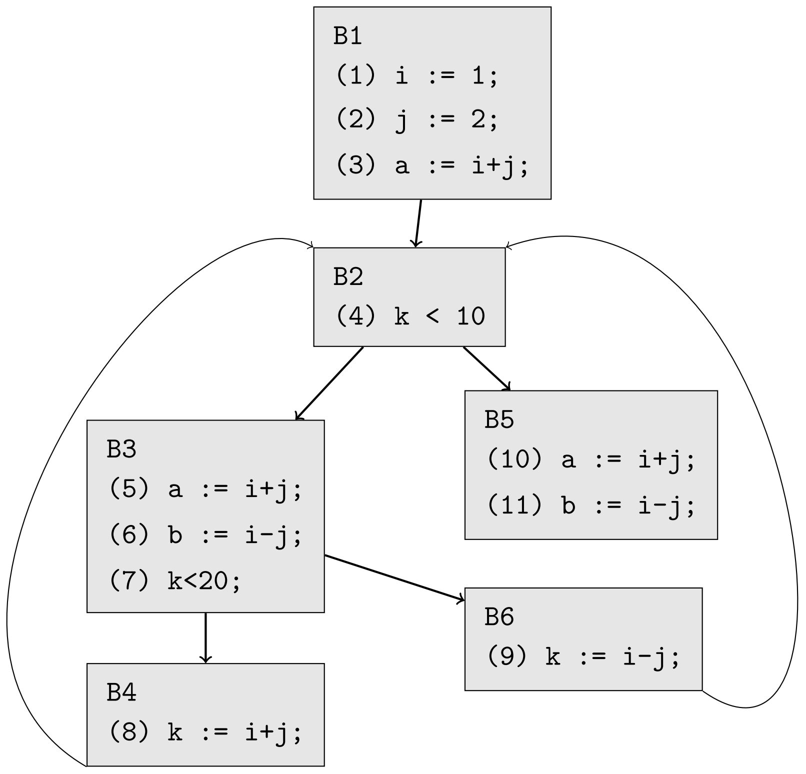
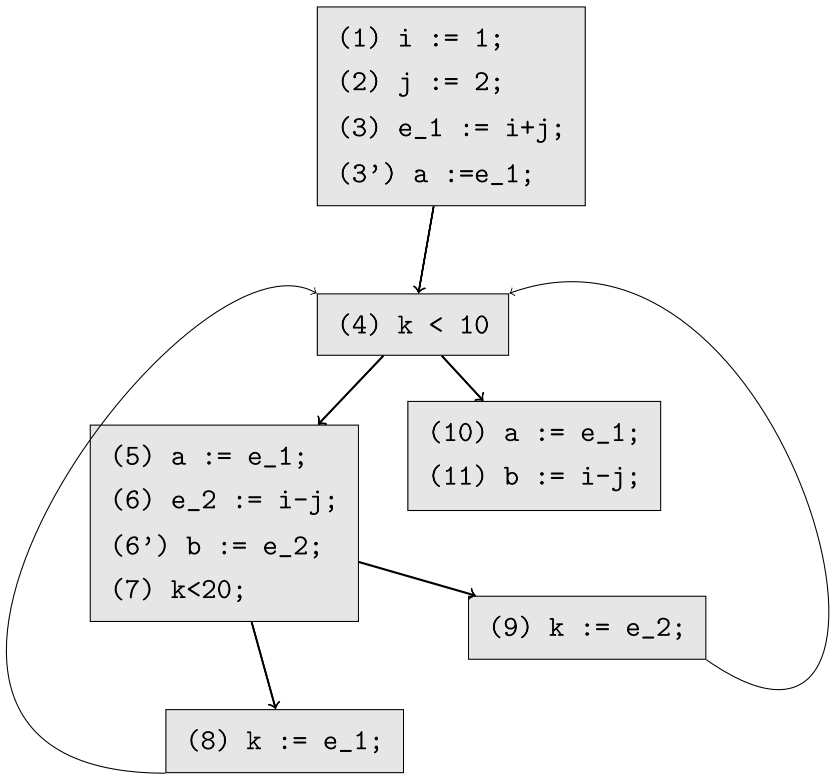


Figure 1: Initial control-flow graph

3. The modified CFG is depicted below.



```
(1) i := 1;
(2) j := 2;
(3) e_1 := i+j;
(3') a :=e_1;
```

```
(4) k < 10
```

```
(5) a := e_1;
(6) e_2 := i-j;
(6') b := e_2;
(7) k<20;
```

```
(10) a := e_1;
(11) b := i-j;
```

```
(9) k := e_2;
```

```
(8) k := e_1;
```