Université Joseph Fourier
MOSIG 1

UFR IM2AG
Academic year 2014-2015

PLCD

Mid-term Exam, November 3, 2014

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.

Exercise 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:

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

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 $(\mathbf{Aexp'} \times \mathbf{State}) \cup (\mathbb{Z} \times \mathbf{State})$ (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.

Exercise 1

1. Give the natural operational semantics of arithmetical expressions by supposing that operands of a binary operator are evaluated from left to right.

2. What is the value of expression $\overset{3}{x} + (\overset{3}{x} + \overset{4}{+})$ where x is initially equal to 3? Check on this example the coherence of the rules given in the previous question.

3. We modify statements by considering the following syntax:

$$S \;:=\; x := a \mid x \mathrel{+}:= a \mid S_1; S_2$$

   Informally, to evaluate statement $x \mathrel{+}:= a$ in memory σ,

   - we first evaluate a, we thus obtain a value v and a new memory σ',
   - then we associate value $v + \sigma(x)$ to x.

   Give the operational semantics for statements $x\ := a$ and $x \mathrel{+}:= a$.

4. Execute $x \mathrel{+}:= ++y$ and $x \mathrel{+}:= ++x$ on $\sigma = [x \mapsto 3, y \mapsto 5]$

5. For each 2-tuple of expressions or statements, indicate whether there are semantically equivalent (prove your answer).

   - $x+ := a$ and $x := x + a$ for $a \in \mathbf{Aexp}'$.
   - $x+ := a$ and $x := a + x$ for $a \in \mathbf{Aexp}'$.

Exercise 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:

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

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: `Pos` $<$ `Int` and `Neg` $<$ `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:

- $Env \times Aexp \cup \{\texttt{Pos}, \texttt{Neg}, \texttt{Int}\}$ for arithmetical expressions,
- $Env \times Bexp \cup \{\texttt{Bool}\}$ 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).

Exercise 2

Give the rules for subtraction and multiplication.

Part B: Boolean Expressions

For boolean expressions, we consider the following syntax:

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

Exercise 3

1. Give the rules for equality
2. Give the rules for inequality.

Exercise III: Optimization (5 points)

We consider the control-flow graph in figure 1 :

Exercise 4

1. Compute the sets $Gen(b)$ and $Kill(b)$, for each basic bloc b.
2. Compute the sets $In(b)$ and $Out(b)$, for each basic block b.
3. Supress redundant computations.

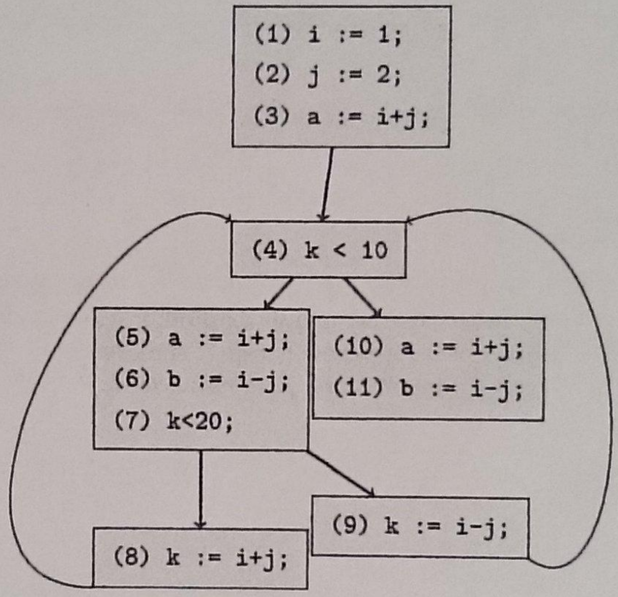

Figure 1: Initial control-flow graph