ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

J. Hoenicke
J. Christ

02.11.2011
Hand in solutions via email to
`christj@informatik.uni-freiburg.de`
until 09.11.2011 (only Java sources and
PDFs accepted)

Tutorials for "Formal methods for Java"
Exercise sheet 2

Exercise 1: Operational semantics

Consider the following Java class:

```
class C {
  private boolean b = true;
  public int m(int x) {
    return (this.b = !this.b) ? ++x : x;
  }
}
```

Give rules for the operational semantics of the `?:` and the `!` operator. Use the rules defining the operational semantics of Java to compute the result of the method call: `c.m(4)`. Assume that `c` is an instance of class `C` which has just been initialized.

Exercise 2: Loops with breaks

Java provides the **break** statement that when executed within a loop causes the execution of the loop to be stopped immediately. Execution is then continued with the first statement after the corresponding loop. For simplicity, we assume every loop is labeled, and every **break** statement is followed by a label, i.e., a **while** loop has the form $l : \mathit{while}(e)s$ where l is the label of the loop.
We can model **break** statements by extending the flow component of program states:

$$\mathit{Flow} ::= \mathit{Norm} | \mathit{Ret} | \mathit{Exc}\langle\langle \mathit{Address} \rangle\rangle | \mathit{Break}\langle\langle \mathit{Label} \rangle\rangle \ .$$

Use this extension to define the operational semantics of **break** l statements and **while** loops with breaks.
Hint: You only need to define two axioms.

Exercise 3: Operational equivalence

We say that two Java statements c_1 and c_2 are operationally equivalent if

$$\begin{aligned} \forall \mathit{flow}, \mathit{heap}, \mathit{lcl}, \mathit{flow}', \mathit{heap}', \mathit{lcl}'.\ & (\mathit{flow}, \mathit{heap}, \mathit{lcl}) \xrightarrow{c_1} (\mathit{flow}', \mathit{heap}', \mathit{lcl}') \iff \\ & (\mathit{flow}, \mathit{heap}, \mathit{lcl}) \xrightarrow{c_2} (\mathit{flow}', \mathit{heap}', \mathit{lcl}') \end{aligned}$$

Are the following pairs of Java statements operationally equivalent? Give a proof or a counter-example.

(a) $y = x\texttt{++;}$ and $y = x\texttt{;}\ x\texttt{++;}$, where x and y are local variables.

(b) $\textbf{if}\,(e)\ c\ \textbf{else}\ c$ and c ,
where e is a boolean expression and c a statement.

(c) $l : \textbf{while}(e)\ c$ and $l : \textbf{while}(\textbf{true})\ \{\textbf{if}(!e)\ \textbf{break}\ l; \textbf{else}\ c\}$,
where l is a label, e a Boolean expression and c a statement (use your rules from Exercise 2).

(Bonus) Try to find a counterexample to the equivalence of $e_1 < e_2$ and $-e_1 > -e_2$ where e_1 and e_2 are integer-valued expressions. Although we did not present a rule for negation, less, and greater you should assume the Java semantics.