Sheet 1: Intra-procedural static analysis

Solutions to this sheet are due on Monday Nov. 10th, 23:59.

All assignments for this course are to be submitted through a version-control system called Subversion. We invite you to read Chapters 1 and 2 of the Subversion Book at:
`http://svnbook.red-bean.com/`

Answer all of the following questions in a PDF file `solution.pdf`. Place this PDF file into your personal folder at `students/your-group/sheet-1`.

For questions, we will be monitoring the discussion forum at:
`https://www.fachschaft.informatik.tu-darmstadt.de/forum//viewforum.php?f=554`
Please ask your questions there.

Aufgabe 1

a) The following code is written in the notation of the smali/baksmali (dis)assembler for Android bytecode. Inform yourself about this representation and, by hand, convert this code into Jimple or into a similar stack-less representation with typed local variables. The file `dalvik-opcodes.html` in SVN gives information about the Dalvik bytecode that Android uses.

```
const/4 v0, 0x0
const/4 v1, 0x1

if-eqz p1, :cond_0

add-int v1, v1, v0

:goto_0
return-void

:cond_0
invoke-virtual {p0, v0}, LMainActivity;->test(Ljava/lang/Object;)I

goto :goto_0
```

2 points

b) Briefly explain a key challenge with this example with respect to typing local variables and how you are addressing this challenge.
1 point

Aufgabe 2

a) Constant propagation is a static analysis which associates at each statement each numeric variable with a numeric constant (where applicable). *Linear* constant propagation is a variant of this analysis which propagates constants for all assignments that form linear equations of the form $y = a \cdot x + b$ where a and b are constants. As an example, for the statement sequence `x = 1; y = 1 + 2 * x;` the analysis would eventually associate `y` with the value 3. For all other equations, the analysis implicitly or implicitly associates the left-hand side with $\top$, representing any "potentially non-constant value". In this part of the exercise, use a sensible notation to define linear constant propagation in terms of (1) analysis direction, i.e., forward or backward, (2) data-flow lattice, (3) flow functions, (4) initialization and (5) merge function. You can neglect assignments to fields or arrays; consider local variables and constants only. Hint: There is no need to explicitly store your $\top$ value; it can be encoded implicitly.
3 points

b) Using the formal notation from above, prove that linear constant propagation is a distributive analysis problem, i.e., for each flow function f and each two in-sets in_1 and in_2 and your merge operator $\sqcap$ show that $f(in_1 \sqcap in_2) = f(in_1) \sqcap f(in_2)$.
4 points

c) Give an example showing that *full* constant propagation is not distributive, i.e., give an appropriate non-linear flow function f and inputs in_1, in_2 such that:
$f(in_1 \sqcap in_2) \neq f(in_1) \sqcap f(in_2)$ or $f(in_1 \sqcap in_2) \not\sqsubseteq f(in_1) \sqcap f(in_2)$
(Be reminded that $f(in_1 \sqcap in_2) \sqsupseteq f(in_1) \sqcap f(in_2)$ always holds in any monotone framework.)
An example of a non-linear equation is $x = y + z$.
2 points

d) On the high level, explain what these results mean with respect to the computability of the MOP solution for these problems.
1 points

Aufgabe 3

Give a simple example of what happens when instantiating the monotone framework with a non-monotone flow function. An informal description suffices. Using the example, show what problems a such non-monotone flow functions can cause.
2 points

Note on plagiarism

If you copy text elements from other sources, clearly mark those elements and state the source. Otherwise, we will have to consider your solution as a case of plagiarism, which will carry severe consequences such as failing this class. Copying solutions from students outside your group is strictly prohibited, is considered plagiarism and will entail the same consequences.