PES

Technische Universität Berlin

Software Engineering for Embedded Systems – Prof. Dr. Sabine Glesner

www.pes.tu-berlin.de Secretariat TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin

Exercise to the Lecture
Analysis and Optimisation of Embedded Systems
SoSe 2015

Prof. Dr. Sabine Glesner Dr. Thomas Göthel

Exercise 11 (graded)

Distribution: June 15, 2015
Delivery: July 02
Discussion: July 03/ July 06, 2015

- Start with this exercise sheet early enough! Ask questions if anything is unclear!
- This is the second of two graded exercise sheets. Please work on it in groups of three students.
- If your former group has fallen apart, please form new groups, for example, using the ISIS discussion forum. Inform Lydia after having formed a new group.
- One group member needs to submit the solution via the ISIS website **by the delivery date**.
- You need 10 of 20 points to pass this exercise sheet.

Problem 1: Implementation of Analyses and Optimisations (8 points)

On the ISIS website, you find the infrastructure code for program analyses in Java. In this exercise, your task is to implement some program analyses and a program optimisation. For the submission, you need to provide the zip-file of the exported project. To this end, right-click on the Project Folder (in Eclipse) and choose *Export* $\longrightarrow$ *General* $\longrightarrow$ *Archive File*. Then, take *PA_grpXX.zip* as the name for the zip file where you replace XX by your corresponding group number. Furthermore, you should provide the test results that are described below as PDF files.

a) Implement the analyses below. For each analysis, provide a corresponding class in the analysis package. Every analysis should either extend the given *Analysis* class (preferred) or it should be based on a copy of the Analysis class with "Domain"

manually replaced accordingly. Have a look on the TODOs in the Analysis class.
Hint: Note that in the constructor of *Analysis*, the flow graph is automatically generated. In this construction, the labels are generated accordingly. How the labelled program looks like can be found out by printing the statement after instantiating the Analysis with this statement. All relevant parts of the flow graph are referenced by attributes of *Analysis*. This is actually all necessary information that you need to define the analyses.

- "LiveVariables" analysis
- "AvailableExpressions" analysis
- "Dominator" analysis

b) Provide test procedures that show that you get the expected results for the programs given in Exercise 5 (Problem 1), Exercise 6, Exercise 7 (Problem 6), and Exercise 9 (Problem 2), respectively. These programs are also contained in the *programs* folder as *program1.txt*, *program2.txt*, *program3.txt*, and *program4.txt*, respectively. For the test procedures, you can adapt *testLive()*, *testAvail()*, and *testDominator()*, respectively, which can be found in *Test.java*.

c) Implement the "ReachingDefinitions" analysis.
Hint: First, think about how you can represent the domain of this analysis. If you choose to implement a new class for the domain, then you should overwrite the *hashCode* and *equals* method. This is necessary to be able to work with *HashMaps* and *HashSets* properly[1]. To do so, right-click on the domain class and follow *Source* $\longrightarrow$ *Generate hashCode() and equals()* $\longrightarrow$ *ok*. Nothing else needs to be done.

d) Provide methods *calcDU()* and *calcUD()* that calculate the *du* and *ud* chains based on the reaching definitions.

e) Implement the constant folding optimisation.
Hint: Use your previously defined *calcDu()* and *calcUD()* methods. Furthermore, the analysis class has the attribute *Statement stmt*. For the optimised version, you might want to copy the original statement using *stmt.copy()*. To replace a variable with name *var* at label l with an arithmetic expression e in a statement s, you might want to use *s.subst(l,var,e)*.

f) Adapt the test methods *testReaching()*, and *testFolding()* and show that you get the expected results for the programs given in Exercise 3 (Problem 3) and Exercise 4 (Problem 1), which are also provided as *program5.txt* and *program6.txt*.

[1]See for example `http://tutorials.jenkov.com/java-collections/hashcode-equals.html`.

Problem 2: Questionnaire: Orders, Lattices (5 points)

Answer whether the following statements hold or not. Provide a brief explanation (if positive) or a counter example (if negative).

a) Every antisymmetric relation can be extended (by only adding tuples to the relation) to a partial order.

b) Every partial order $(L, \sqsubseteq)$ is a lattice.

c) The the empty set together with the empty relation form a lattice.

d) Every greatest element is a maximal element.

e) Every maximal element is a greatest element.

f) For every set L, it is possible to define a relation $\sqsubseteq$ such that $(L, \sqsubseteq)$ is a complete lattice.

g) Every non-empty finite lattice is complete.

h) Every complete lattice has a greatest and a least element.

i) Every monotone function defined on a lattice has a least fixed point.

j) Every monotone function defined on a complete lattice has a least fixed point.

Problem 3: Monotone Frameworks (7 points)

Answer the following questions.

a) What is a monotone framework?

b) What is an instance of a monotone framework? Explain it by using the Constant Propagation Analysis as an example.

c) Perform the Constant Propagation Analysis for the following program.

$[a := 0]^1 \ ; \ [b := 10]^2 \ ; \ [k := 2]^3 \ ; \ \mathit{while} \ [a < b]^4 \ \mathit{do} \ [a := a + k]^5 \ \mathit{od} \ ; \ [b := a - k]^6$

d) What is the difference between MOP and MFP? Discuss the precise equations, name advantages and disadvantages.

Problem 4: Value Analysis (not graded - only discussion in class)

In the lecture, the *Value Analysis* as an alternative to the *Constant Propagation Analysis* was presented. It is based on the lattice $State_{\text{VA}} := ((Var_* \to \mathbb{I})_\perp, \sqsubseteq)$, with $\mathbb{I} := \{[l,u] | l \in \mathbb{Z} \cup \{-\infty\}, u \in \mathbb{Z} \cup \{\infty\}, l \leq u\}$. The partial order on $State_{VA}$ is defined as $[l_1,u_1] \sqsubseteq [l_2,u_2] \Leftrightarrow l_2 \leq l_1 \wedge u_1 \sqsubseteq u_2$. The supremum is consequently defined as $[l_1,u_1] \sqcup [l_2,u_2] := [\min(l_1,l_2), \max(u_1,u_2)]$. The addition on intervals is defined as

$$[l_1,u_1] + [l_2,u_2] := [l_1+l_2, u_1+u_2], \quad -\infty + x = -\infty, \ \infty + x = \infty$$

a) What is the goal of the Value Analysis? What are the advantages?

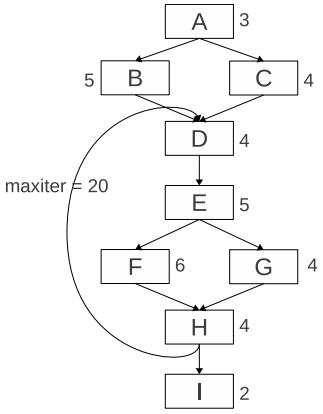
b) Categorise the analysis (forward, backward, must, may).

c) Perform the analysis and provide the entry and exit solution for the following program:

$$[a := 0]^1; [b := 10]^2; [k := 2]^3; \ \mathit{while}\ [a < b]^4\ ([a := a + k]^5); [b := a - k]^6$$

Problem 5: WCET Bound Calculation (not graded - only discussion in class)

Given the following control flow graph with local WCET annotations.



Calculate the estimation of the WCET by applying the following approaches:

a) path-based

b) structure-based