Carl von Ossietzky
Universität
Oldenburg

Using CPAchecker in Teaching
An experience report

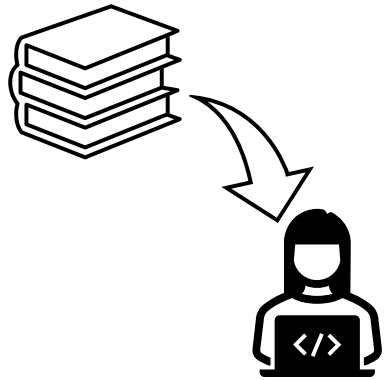
By Jan Haltermann

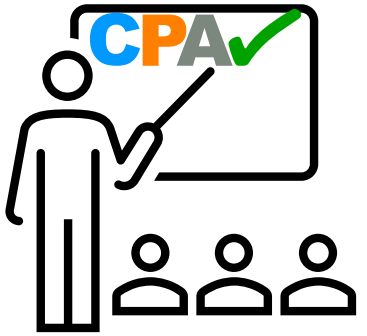
CPAchecker-Workshop 2023

Where and why do we use CPAchecker

- Bachelor: “Practical course in Software Verification”
- Master: “Software Analysis”
- Lecture by Heike Wehrheim, supported by 1 or 2 PhD Students
- Within bachelor and master theses [Not topic of this talk]

- Overall Goals:
  - Bridge between theory and practice
  - Teach students how to use CPAchecker





Software Analysis Overview

Organization (6 ECTS)

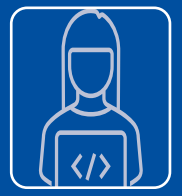
- 2/3 Lecture, 1/3 Labs

Participants:

- Approx. 25 Students

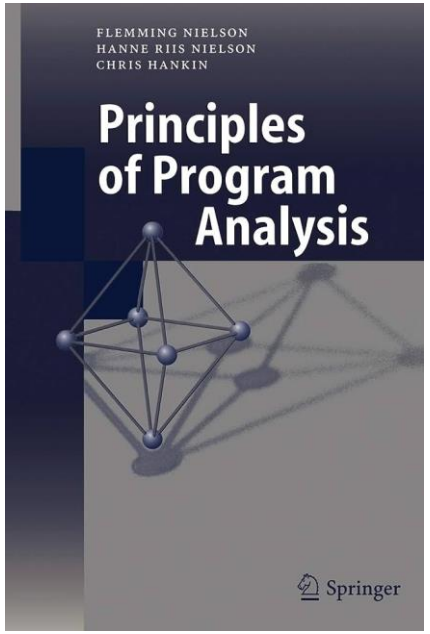
Lecture

- Concepts of Software analysis

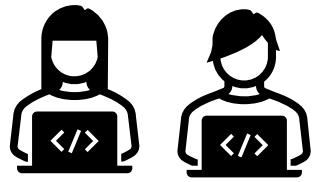


Labs:

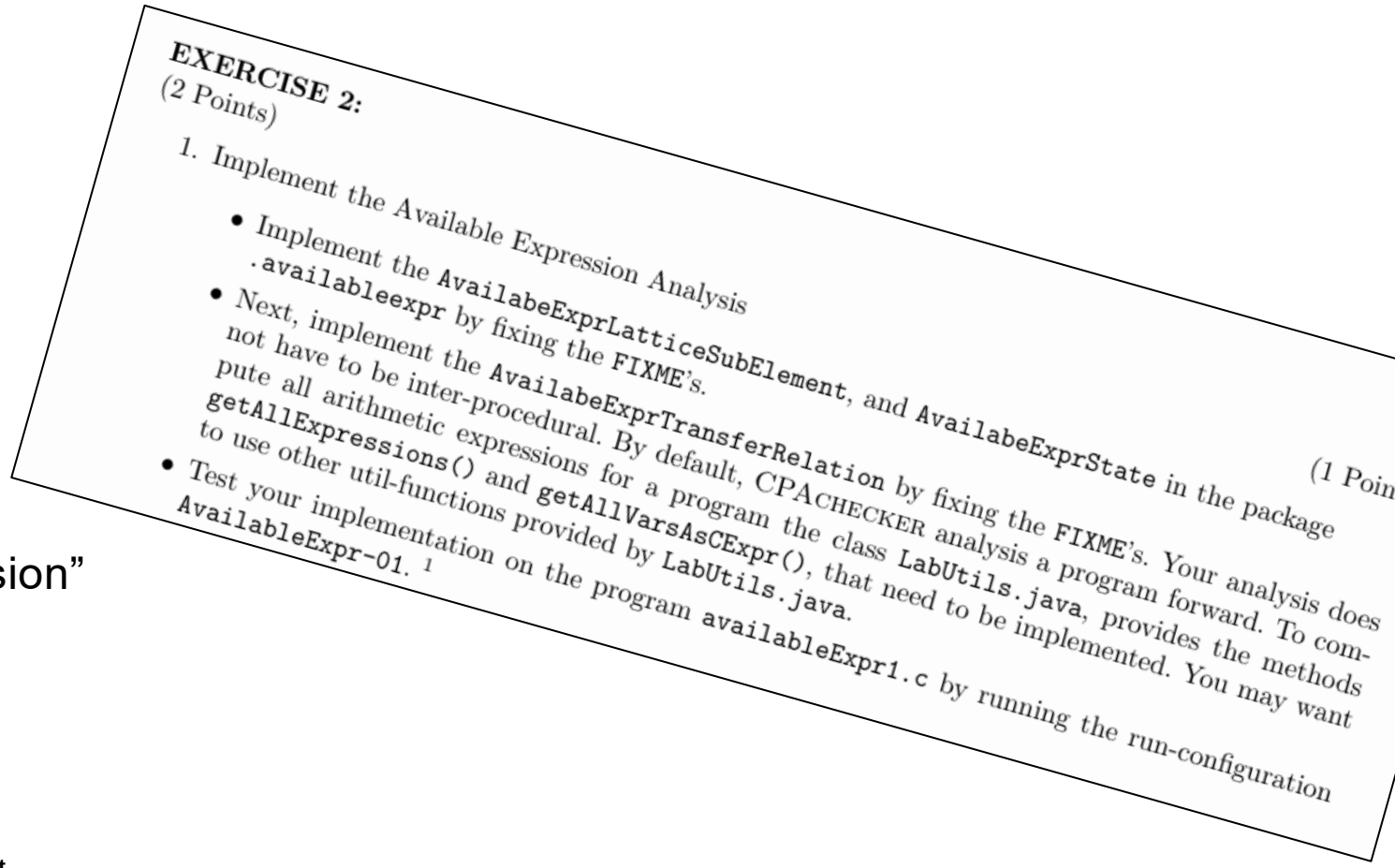
- Implement analyses from lecture



Task for the Labs



- Implement up to five different analyses:
  - Reaching Definitions
  - Available Expressions
  - Taint analysis
  - Constant propagation
  - Interval analysis
- Tasks corrected in “discussion”

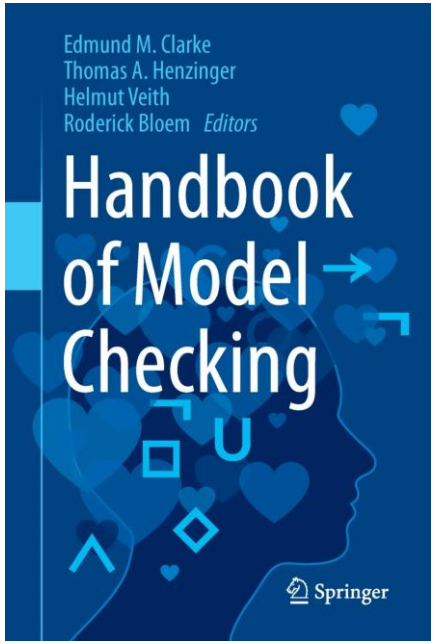


EXERCISE 2:
(2 Points)

1. Implement the Available Expression Analysis (1 Poin

   - Implement the `AvailabeExprLatticeSubElement`, and `AvailabeExprState` in the package `.availableexpr` by fixing the `FIXME`'s.
   - Next, implement the `AvailabeExprTransferRelation` by fixing the `FIXME`'s. Your analysis does not have to be inter-procedural. By default, CPAchecker analysis a program forward. To compute all arithmetic expressions for a program the class `LabUtils.java`, provides the methods `getAllExpressions()` and `getAllVarsAsCExpr()`, that need to be implemented. You may want to use other util-functions provided by `LabUtils.java`.
   - Test your implementation on the program `availableExpr1.c` by running the run-configuration `AvailableExpr-01`. 1

“Practical course in Software Verification“ Overview



Organization (6 ECTS)

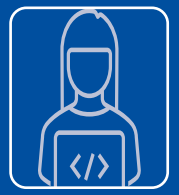
- 1/3 Lecture, 2/3 Group work

Participants:

- 9 Students (max 12), 3 per group

Lecture

- Theoretical background of CPAs



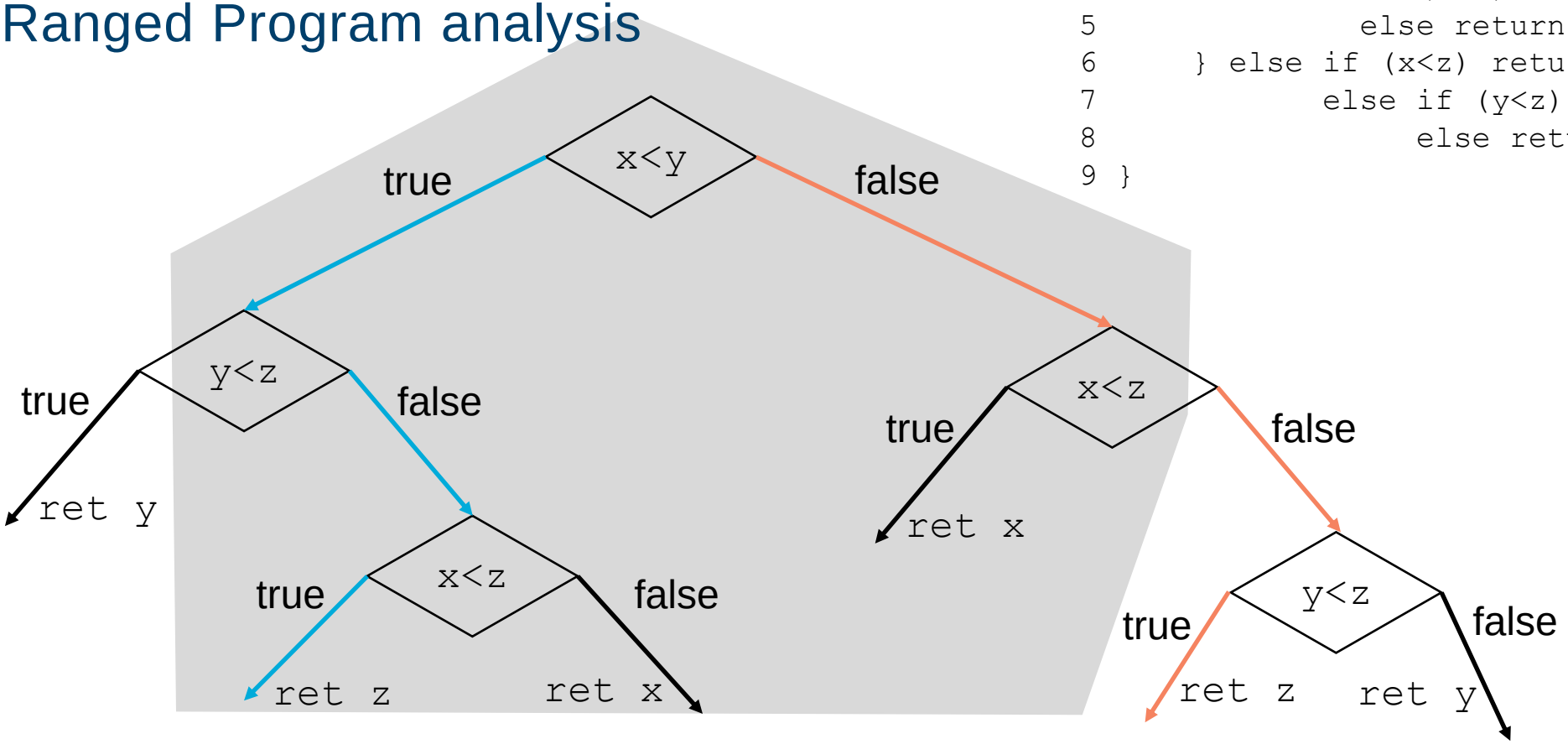
Group work:

- Implement ranged program analysis

Carl von Ossietzky
Universität
Oldenburg

Task for the Group Work: Ranged Program analysis

```
1 int mid(int x, int y, int z) {
2       if (x < y) {
3          if (y<z) return y;
4          else if (x<z) return z;
5               else return x;
6      } else if (x<z) return x;
7             else if (y<z) return z;
8                  else return y;
9 }
```

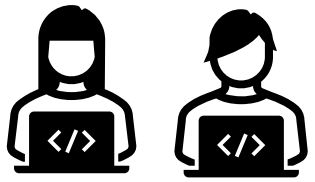


Each path can be described by concrete Inputs:

[$x \mapsto 3, y \mapsto 8, z \mapsto 5$; $x \mapsto 42, y \mapsto, 112\ z \mapsto 1337$)

Carl von Ossietzky
Universität
Oldenburg

Task for the Group Work: Ranged Program analysis



- Work divided in three phases
  - Phase 1: Implement constant propagation (with concrete inputs)
  - Phase 2: Composition of constant propagation and symbolic execution
  - Phase 3: Evaluate compositions using different analyses
- Introduction for each phase with talk

What did we build in the CPAchecker

- Skeleton for each analysis
  - CPA, TransferRelation and State for each analysis

```
private ConstantPropagationState handleDeclarationEdge(
    ConstantPropagationState pState, CDeclarationEdge pCfaEdge, CDeclaration pDeclaration) {
  // FIXME: Implement this

  return new ConstantPropagationState(pState);
}
```

  - A lot of util methods
- Introduction to CPAchecker
  - General usage, structure of CStatement and CExpression, how to debug, …
- A VM with the project

Practical course in Software Verification: Students Feedback

- Students liked the idea:
  - Lecture got good grades in evaluation
  - Liked the mix of theory and practice
  - Liked the idea of “competition”

“meaningful & realistic tasks”

  - But we observe some general “problems”

General Observations

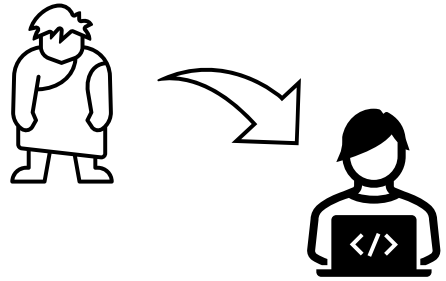
"CPAchecker does not behave as expected"

"My code does not compile as it should"

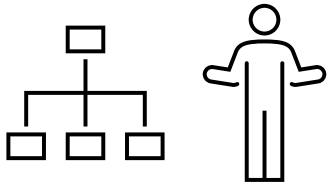
General Observations

General Observations

- Students became faster in solving tasks



- The type hierarchy is complex to understand



- Information provided get lost

“How to get all variables in a CExpression?”

```
/**
 * Computes the set of variables present in the given expression
 *
 * @param pExpression expression to process
 * @return a set of the variable names present
 */
6 usages
public static Set<String> getVarsPresent(CExpression pExpression) {
```

Software Analysis: Short Survey

	agree		neutral		disagree
– Tasks were easy to understand:	○	●	○	○	○
– The tasks where too complex:	○	○	●	○	○
– Developing in CPAchecker is difficult:	○	●	○	○	○
– Eventually I understood CPAchecker’s structure:	○	●	○	○	○
– The lab helps to understand the lecture:	●	○	○	○	○

Carl von Ossietzky
Universität
Oldenburg

Summary and open questions

- General: Hands-on experience is liked
- Some students liked CPAchecker and started working in the group afterwards
- Some other asked for thesis explicitly not with CPAchecker

Open questions:

- How to provide information in introduction to CPAchecker
  - All at once → Information and students get lost
  - Step by step if needed → Students wait and hard to schedule
- How to introduce structure of CStatement and CExpression