



Seminar *Trends in Computer-Aided Verification*

Introduction

Winter Semester 2022; October 18, 2022

Thomas Noll et al.

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ws-22-23/cav/>

Outline

Overview

Aims of this Seminar

Important Dates

A. Verification of Neural Networks [Christopher Brix]

B. Static Program Analysis for Software Verification [Thomas Noll]

Final Hints

Formal verification methods

- Rigorous, mathematically based techniques for the specification, development and verification of software and hardware systems
- Aim at improving correctness, reliability and robustness of such systems

Formal verification methods

- **Rigorous, mathematically based techniques** for the specification, development and verification of software and hardware systems
- Aim at improving **correctness, reliability and robustness** of such systems

Classifications

- According to **design phase**
 - specification, implementation, testing, ...
- According to **specification formalism**
 - source code, neural networks, Bayesian networks, fault trees, ...
- According to underlying **mathematical theories**
 - model checking, theorem proving, static analysis, ...

Outline

Overview

Aims of this Seminar

Important Dates

A. Verification of Neural Networks [Christopher Brix]

B. Static Program Analysis for Software Verification [Thomas Noll]

Final Hints

Goals

Aims of this seminar

- **Independent understanding** of a scientific topic
- Acquiring, reading and understanding **scientific literature**
 - given references sufficient in most cases
- Writing of your **own report** on this topic
 - far more than just a translation/rewording
 - usually an “**extended subset**” of original literature
 - “subset”: present core ideas and omit too specific details (e.g., related work or optimisations)
 - “extended”: more extensive explanations, examples, ...
 - discuss contents with supervisor!
- **Oral presentation** of your results
 - can be “proper subset” of report
 - generally: less (detailed) definitions/proofs and more examples

Requirements on Report

Your report

- Independent writing of a report of **12–15 pages**
- First milestone: **detailed outline**
 - not: “1. Introduction/2. Main part/3. Conclusions”
 - rather: overview of structure (section headers, main definitions/theorems) and initial part of main section (one page)
- **Complete** set of references to all consulted literature
- **Correct citation** of important literature
- **Plagiarism**: taking text blocks (from literature or web) without source indication causes immediate **exclusion from this seminar**
- Font size **12pt** with “standard” page layout
 - **L^AT_EX template** will be made available on seminar web page
- **Language**: German or English
- We expect the **correct usage** of spelling and grammar
 - ≥ 10 errors per page $\implies$ abortion of correction

Requirements on Talk

Your talk

- Talk of **30 minutes**
- Available: projector, presenter, [laptop]
- Focus your talk on the **audience**
- **Descriptive** slides:
 - $\leq$ 15 lines of text
 - use (base) colors in a useful manner
 - number your slides
- **Language:** German or English
- No spelling mistakes please!
- Finish **in time**. Overtime is bad
- Ask for **questions**
- Have **backup slides** ready for expected questions
- **L^AT_EX/beamer template** will be made available on seminar web page

General Information

This document provides various criteria used to assess the performance of seminar participants. There are four milestones, namely

- detailed outline,
- full report,
- presentation slides, and
- seminar talk.

Each milestone has its own set of requirements which are evaluated during the semester—allowing students to get intermediate feedback on their performance. A certain amount of points is granted for each fulfilled requirement. If all requirements are fulfilled (i.e. not **failed**), the final grade for the course is given by the sum of the collected points using the following table:

Points:	≥ 30	≥ 33	≥ 36	≥ 39	≥ 42	≥ 45	≥ 48	≥ 51	≥ 54	≥ 57
Grade:	4.0	3.7	3.3	3.0	2.7	2.3	2.0	1.7	1.3	1.0

In exceptional cases, we reserve the right to deviate from this grading scheme (e.g. bonus points for going above and beyond). Moreover, **meeting the deadlines is a must!** If the deadline is missed by no more than one day, 3 points are deducted. Missing a deadline by more than one day (i.e. more than 24 hours) will lead to the seminar to be considered “not passed”. In very exceptional cases (such as being ill for a longer period), the deadline may be extended, please contact us proactively!

Hint 1: Look at our website for additional material on how to set up good reports/presentations.

Hint 2: Communicate potential issues with your supervisor as soon as possible.

Student Name: _____

Detailed Outline

10 Points

The goal for this milestone is to provide an overview of the overall structure (section headers, main definitions/theorems/examples) as well as an initial part of one of the main sections of the report. When the detailed outline is due, students are expected to be familiar with their topic and the associated literature.

Deadline: Please refer to our website.

Overview of Structure (4 Points)

Score: _____

The future contents of the individual (sub-)sections should become clear to the supervisor. If necessary, short bullet points can be added for clarification.

Initial Content (3 Points)

Score: _____

There should be at least one page of actual content for one of the main sections.

Language (3 Points)

Score: _____

The report can be in German or English. We expect proper scientific writing and the correct usage of spelling and grammar.

Student Name: _____

Full Report

20 Points

By the deadline for this milestone, the full report should be finished and proofread multiple times.

Deadline for the version we give feedback on: Please refer to our website.

Deadline for the final version: Please refer to our website.

Mathematical Notation (2 Points)

Score: _____

Mathematical notation should be used wherever necessary to provide clear, well-defined descriptions. It should be consistent both within the report and with other literature.

Document Layout / Typesetting (2 Points)

Score: _____

Definitions and equations must be readable, using suitable L^AT_EX environments. Bullet points must be appropriately used, cross-references to other parts of the report should be correct and used in a reasonable manner. Figures should not distract from other content.

Understandability (4 Points)

Score: _____

The content of the source material should be explained in the report in such a way that no prior knowledge of the source material is necessary to understand the report. Non-trivial prerequisites that are not explained in the report should be clearly stated in the report and communicated to the supervisor in advance.

Examples (2 Points)

Score: _____

Examples should be used to demonstrate the general theories. If possible, a single running example is better than many individual ones and a custom example is better than one copied from the source material.

Amount of Content (2 Points)

Score: _____

The report should have 12–15 pages. Significant deviations (e.g. an additional appendix with figures) should be discussed with the supervisor.

Hint: Do not stretch the number of pages with, e.g., unnecessary page breaks, weird figure placements, “empty phrases”, etc. The pages that were gained by such techniques do not count.

Complete and Correct Citation (2 Points)

Score: _____

All used literature must be cited correctly. This includes giving citations for all copied figures, tables, etc.

Supervisor Feedback (3 Points)

Score: _____

All the feedback from the supervisor (both for the detailed outline and the full report) needs to be addressed adequately. In case there is a reason to not implement certain feedback (e.g. follow-up questions, further doubts), it shall be discussed with the supervisor.

Language (3 Points)

Score: _____

The report can be in German or English. We expect proper scientific writing and the correct usage of spelling and grammar.

Student Name: _____

Presentation Slides

10 Points

In this milestone the presentation slides are due. You may use a different language (German or English) than the report. It is also fine to e.g. present in German with English slides.

Deadline: Please refer to our website.

Structure (2 Points)

Score: _____

There should be a clear thread running through the presentation. The different sections should build upon each other, leading to a clear conclusion. Relevant definitions are given before their usage, but are introduced as late as possible to avoid mental overload.

Layout of Slides (4 Points)

Score: _____

Slides should not be used to convey all the information, but to support the oral presentation. They are not a script, so they should be simpler than what is used during lectures.

Level of Detail (4 Points)

Score: _____

Explanations can be example-driven. The level of detail should be adapted to the audience (i.e. other participating students).

Student Name: _____

Seminar Talk

20 Points

The seminar talk is the last milestone in which the acquired knowledge on the topic should be presented to the audience.

Timing (2 Points)

Score: _____

The time limit should not be exceeded. Students should manage their time by themselves.

Preparation (2 Points)

Score: _____

The speaker should be prepared for giving the presentation. They should know the content and order of their slides.

Spoken Language / Style of Presentation (2 Points)

Score: _____

The audience should understand what the speaker is saying on a language level. The speaker should speak freely and convey a certain confidence in the presentation.

Understandability (6 Points)

Score: _____

It should be possible for the audience to follow the provided explanations without prior knowledge of the report or source material.

Questions and Answers (8 Points)

Score: _____

Questions may be asked by other students and/or by attending supervisors during or after the talk (if they are asked during the presentation, the time limit is extended accordingly). The questions should be answered appropriately and in an understandable manner.

Hint 1: Some questions might be outside the scope of the topic. Sometimes an appropriate answer to those questions is a simple “I don’t know”—followed by an educated guess.

Hint 2: Questions are usually asked in good faith out of interest in your topic, i.e. they should not be confused with an oral examination. In particular, students should not hesitate from asking questions to other students. The affect of student questions on the overall grade is negligible.

Outline

Overview

Aims of this Seminar

Important Dates

A. Verification of Neural Networks [Christopher Brix]

B. Static Program Analysis for Software Verification [Thomas Noll]

Final Hints

Important Dates

Deadlines

- October 21: Topic preferences due
- November 21: Detailed outline due
- December 19: Full report due
- January 16: Presentation slides due
- January 23–25 (?): Seminar talks

Important

Missing a deadline causes **immediate exclusion** from the seminar

Selecting Your Topic

Procedure

- You obtain(ed) a list of topics of this seminar.
- Classified according to BSc/MSc level.
 - MSc students please choose at least one “M-only” topic
- Indicate the preference of your topics (first, second, third).
- Return sheet here or via e-mail (noll@cs.rwth-aachen.de) or to secretary **by Friday (October 21)**.
- We do our best to find an adequate topic-student assignment.
 - disclaimer: no guarantee for an optimal solution
- Assignment will be published on web site next week.
- Then also your **supervisor** will be indicated.
- Please give language preference (unsure $\implies$ German).

Withdrawal

- You have up to **three weeks** to refrain from participating in this seminar.
- Later cancellation (by you or by us) causes a **not passed** for this seminar and reduces your (three) possibilities by one.

Outline

Overview

Aims of this Seminar

Important Dates

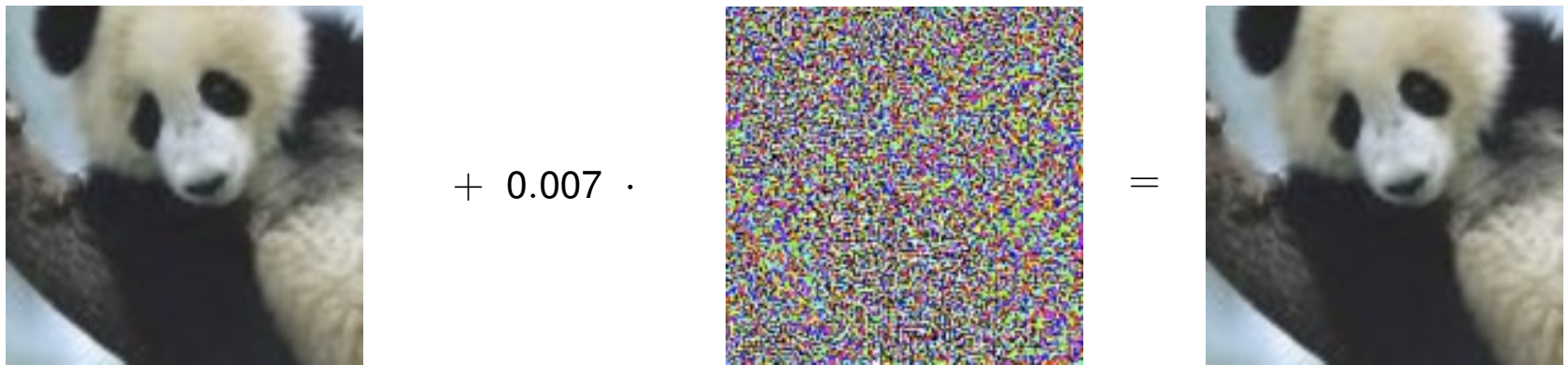
A. Verification of Neural Networks [Christopher Brix]

B. Static Program Analysis for Software Verification [Thomas Noll]

Final Hints

Motivation

- Increasing use of neural networks in safety-critical applications
- Behaviour of trained networks difficult to predict
- Adversarial examples cause (undetected) misclassifications



Adversarial Example [Goodfellow et al., 2015]

1. Verification Using the Simplex Algorithm

- Encoding of the verification problem as a simplex problem
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 67–81

Verification of Neural Networks

1. Verification Using the Simplex Algorithm

- Encoding of the verification problem as a simplex problem
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 67–81

2. Abstraction-Based Verification with Intervals and Zonotopes

- Network behaviour needs to be approximated
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 83–108

Verification of Neural Networks

1. Verification Using the Simplex Algorithm

- Encoding of the verification problem as a simplex problem
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 67–81

2. Abstraction-Based Verification with Intervals and Zonotopes

- Network behaviour needs to be approximated
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 83–108

3. Shared Certificates for Neural Network Verification

- The verification of one (robustness) property can be reused to help proving another one
- Marc Fischer, Christian Sprecher, Dimitar I. Dimitrov, Gagandeep Singh, Martin Vechev: *Shared Certificates for Neural Network Verification*

Verification of Neural Networks

1. Verification Using the Simplex Algorithm

- Encoding of the verification problem as a simplex problem
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 67–81

2. Abstraction-Based Verification with Intervals and Zonotopes

- Network behaviour needs to be approximated
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 83–108

3. Shared Certificates for Neural Network Verification

- The verification of one (robustness) property can be reused to help proving another one
- Marc Fischer, Christian Sprecher, Dimitar I. Dimitrov, Gagandeep Singh, Martin Vechev: *Shared Certificates for Neural Network Verification*

4. Detecting Novel Inputs

- Networks guess: after training on animals, it may return “cat” for cars
- Problem: Identify inputs that are outside the training domain (“don’t know”)
- Thomas A. Henzinger, Anna Lukina, Christian Schilling: *Outside the Box: Abstraction-Based Monitoring of Neural Networks*

Verification of Neural Networks

1. Verification Using the Simplex Algorithm

- Encoding of the verification problem as a simplex problem
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 67–81

2. Abstraction-Based Verification with Intervals and Zonotopes

- Network behaviour needs to be approximated
- Aws Albarghouthi: *Introduction to Neural Network Verification*, pp. 83–108

3. Shared Certificates for Neural Network Verification

- The verification of one (robustness) property can be reused to help proving another one
- Marc Fischer, Christian Sprecher, Dimitar I. Dimitrov, Gagandeep Singh, Martin Vechev: *Shared Certificates for Neural Network Verification*

4. Detecting Novel Inputs

- Networks guess: after training on animals, it may return “cat” for cars
- Problem: Identify inputs that are outside the training domain (“don’t know”)
- Thomas A. Henzinger, Anna Lukina, Christian Schilling: *Outside the Box: Abstraction-Based Monitoring of Neural Networks*

5. Tricking Novelty Detection Techniques

- If a program can identify inputs that are outside the training domain, can we fool this program?
- Sara Hajj Ibrahim, Mohamed Nassar: *Hack The Box: Fooling Deep Learning Abstraction-Based Monitors*

Outline

Overview

Aims of this Seminar

Important Dates

A. Verification of Neural Networks [Christopher Brix]

B. Static Program Analysis for Software Verification [Thomas Noll]

Final Hints

Motivation

Static (Program) Analysis

Static analysis is a general method for **automated reasoning** on artefacts such as requirements, design models, and **programs**.

Motivation

Static (Program) Analysis

Static analysis is a general method for **automated reasoning** on artefacts such as requirements, design models, and **programs**.

Distinguishing features

Static: based on source code, not on (dynamic) execution

- in contrast to testing, profiling, or run-time verification

Automated: “push-button” technology, i.e., little user intervention

- in contrast to interactive “theorem-proving” approaches

Motivation

Static (Program) Analysis

Static analysis is a general method for **automated reasoning** on artefacts such as requirements, design models, and **programs**.

Distinguishing features

Static: based on source code, not on (dynamic) execution

- in contrast to testing, profiling, or run-time verification

Automated: “push-button” technology, i.e., little user intervention

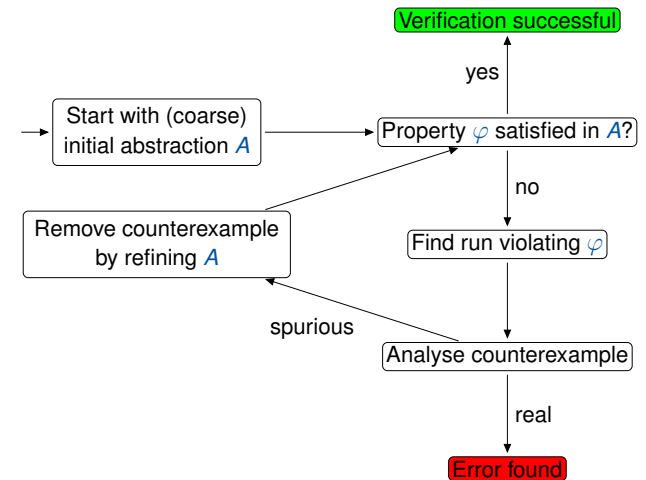
- in contrast to interactive “theorem-proving” approaches

(Main) Applications

- Initially (since 1970s): **compiler optimisations** and synthesis of efficient code
- Now: support for all phases of software development
 - verification of specifications
 - **verification of program correctness**
 - certification of critical software
 - refactoring and maintenance of applications, ...

Predicate Abstraction (SLAM Tool)

- Implements CounterExample-Guided Abstraction Refinement (CEGAR) approach
- Checks behavioural requirements of software interfaces
 - e.g., “a thread may not acquire a lock it has already acquired, or release a lock it does not hold”
- Supports recursive procedures, pointers, and memory allocation
- Sub-tools:
 - C2bp: C program $\times$ Predicates $\rightarrow$ Boolean program (Boolean variables = predicates)
 - BEPOP: symbolic (BDD-based) model checker for (recursive) Boolean programs
 - newton: abstraction refinement
- Developed into commercial product (Static Driver Verifier – SDV; part of Windows Driver Foundation development kit)



Sacrificing Soundness (FindBugs/SpotBugs Tool)

- Goal: detect memory access errors (null pointer dereferences, ...)
- Sophisticated (and costly) analysis techniques exist
- Here: simple non-standard analysis which is neither complete nor sound
 - false negative rates of 20%–50%
 - false positive rates of around 20%



Coverity Tool

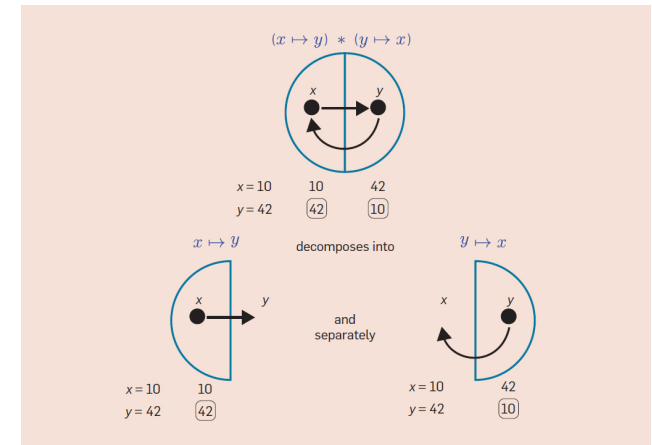
- Introduces extension language for specifying analyses and efficient algorithms for executing these extensions
- Idea: check by compiler preprocessor
- Turned into commercial product



<https://en.wikipedia.org/w/index.php?curid=69721584>

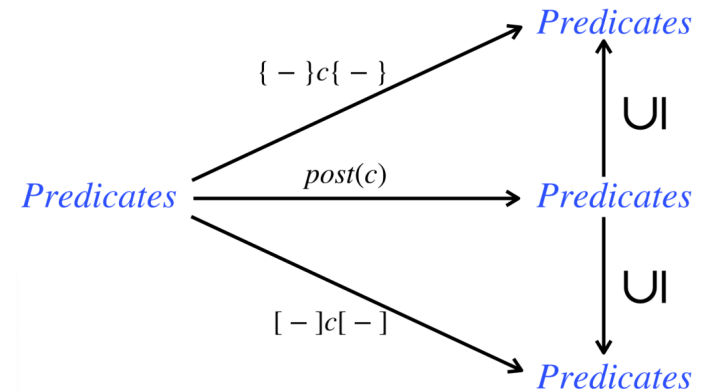
Separation Logic (Infer Tool)

- Extension of Hoare Logic for analysing heap-manipulating programs
- Implemented in Facebook's INFER static analyzer
- Processes codebases with dozens of MLOC



Incorrectness Logic (Pulse-X Tool)

- Dual approach to Hoare Logic
- Aim: defining new automated bug catching techniques by proving presence of errors



```
Hello.java
1  class Hello
2  {
3      int test()
4      {
5          String s = null;
6          return s.length();
7      }
8  }
```

```
Command Line

bash-3.2$ javac Hello.java

error: NULL_DEREFERENCE
       object s last assigned on line 5 could be null
       and is dereferenced at line 6
```

Outline

Overview

Aims of this Seminar

Important Dates

A. Verification of Neural Networks [Christopher Brix]

B. Static Program Analysis for Software Verification [Thomas Noll]

Final Hints

Some Final Hints

Hints

- Take your time to **understand** your literature.
- Be **proactive**! Look for **additional** literature and information.
- Discuss the content of your report with other students.
- Be **proactive**! Contact your supervisor **on time**.
- **Prepare** the meeting(s) with your supervisor.
- Forget the idea that you can prepare a talk in a day or two.

We wish you success and look forward to an enjoyable and high-quality seminar!