Seminar *Advanced Topics in Formal Semantics*

Introduction

Summer 2025; April 14, 2025

Thomas Noll et al.
Software Modeling and Verification Group
RWTH Aachen University

`https://moves.rwth-aachen.de/teaching/ss-25/semantics/`

Chair of
Software Modeling
and Verification
(Computer Science 2)

RWTH AACHEN UNIVERSITY

Outline

Overview

Aims of this Seminar

Important Dates

The Topics

Final Hints

Aspects of Programming Languages

Syntax: “How does a program look like?”

- hierarchical composition of programs from structural components

⇒ *Compiler Construction*

Aspects of Programming Languages

Syntax: “How does a program look like?”

- hierarchical composition of programs from structural components

⇒ *Compiler Construction*

Semantics: “What does this program mean?”

- output/behaviour/... in dependence of input/environment/...

⇒ this seminar

Aspects of Programming Languages

Syntax: “How does a program look like?”

- hierarchical composition of programs from structural components

⇒ *Compiler Construction*

Semantics: “What does this program mean?”

- output/behaviour/... in dependence of input/environment/...

⇒ this seminar

Pragmatics: “Is the programming language practically usable?”

- length and understandability of programs
- learnability of programming language
- appropriateness for specific applications, ...

⇒ *Software Engineering*

Motivation

Main applications

- Implementation of programming languages and algorithms
  - Exact understanding of semantics avoids uncertainties and enables correctness proofs.
- Formal verification methods (here)
  - 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.

Motivation

Main applications

- Implementation of programming languages and algorithms
  - Exact understanding of semantics avoids uncertainties and enables correctness proofs.
- Formal verification methods (here)
  - 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.

(Complementary) Kinds of Formal Semantics

Operational: describes execution of the program on some (very) abstract machine

Denotational: mathematical definition of input/output relation

Axiomatic: formalisation of special properties of programs by logical formulae abstract machine

Areas Covered in this Seminar

Topic areas

- Hoare Logic (axiomatic)
- Separation Logic (operational, axiomatic)
- Software Verification (operational, axiomatic)
- Static Analysis of Quantum Programs (operational)

Outline

Overview

Aims of this Seminar

Important Dates

The Topics

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 that 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 or AI tools) without source indication causes immediate exclusion from this seminar
- Font size 12pt with “standard” page layout
  - LaTeX template will be made available on seminar web page
- Language: German or English
- We expect the correct usage of spelling and grammar
  - $\geq$ 10 errors per page $\Longrightarrow$ 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
  - LaTeX/beamer template will be made available on seminar web page
- Language: German or English
- No spelling mistakes please!
- Finish in time. Overtime is bad
- Ask for questions
- Have backup slides ready for expected questions

Outline

Overview

Aims of this Seminar

Important Dates

The Topics

Final Hints

Important Dates

Deadlines

- April 22: Topic preferences due
- May 19: Detailed outline due
- June 16: Full report due
- June 30: Presentation slides due
- July 14 (?): 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.
- Indicate the preference of your topics (first, second, third).
- Return sheet here or via e-mail (`noll@cs.rwth-aachen.de`) by Tuesday next week (April 22).
- 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 mid next week.
- Then also your supervisor will be indicated.

Withdrawal

- You have up to one week (!) to refrain from participating in this seminar (after topic assignment).
- 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

The Topics

Final Hints

A. Hoare Logic

1. T. Nipkow: Hoare logics for recursive procedures and unbounded nondeterminism, CSL 2002
   - Hoare logics for partial and total correctness of recursive parameterless procedures in the context of unbounded nondeterminism

2. P. W. O'Hearn: Incorrectness Logic, POPL 2020
   - sound techniques for reasoning about the *presence* of bugs

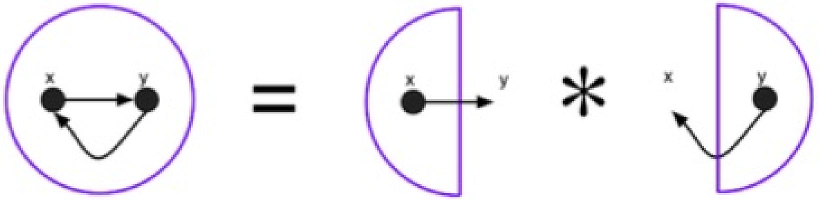
3. N. Zilberstein, D. Dreyer, A. Silva: Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning, OOPSLA 2023
   - a unified theory for reasoning about both correctness (classical Hoare Logic) and incorrectness (Incorrectness Logic)

4. L. Verscht, B. L. Kaminski: A Taxonomy of Hoare-Like Logics: Towards a Holistic View using Predicate Transformers and Kleene Algebras with Top and Tests, POPL 2025
   - a taxonomy of different program logics considering aspects like program termination, determinism, and reversibility

$$\text{(seq)}\ \frac{\{A\}\ c_1\ \{C\} \qquad \{C\}\ c_2\ \{B\}}{\{A\}\ c_1; c_2\ \{B\}}$$

B. Separation Logic

1. P. W. O'Hearn: A Primer on Separation Logic (and Automatic Program Verification and Analysis), Software Safety and Security, 2012
   - gentle introduction to the topic (lecture notes)
2. A. A. de Amorim, C. Hritcu, B. C. Pierce: The Meaning of Memory Safety, POST 2018
   - characterisation of memory safety to support local reasoning about state (non-interference properties, frame rule)
3. V. Vafeiadis: Concurrent Separation Logic and Operational Semantics, ENTCS 276 (2011)
   - extension of SL to concurrent threads (CSL) and soundness proof based on operational semantics

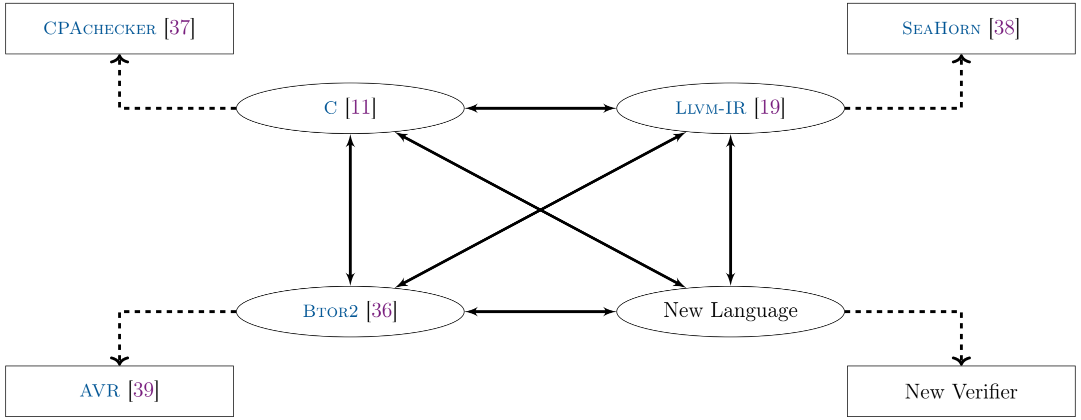


$$\frac{\{A\}\, c\, \{B\} \qquad FV(C) \cap Mod(c) = \emptyset}{\{A * C\}\, c\, \{B * C\}}$$

C. Software Verification

1. R. Majumdar, V. R. Sathiyanarayana: Sound and Complete Techniques for Reasoning About Termination, Springer LNCS 15260, 2025
   - overview of termination problems and related analysis methods for a variety of Turing-complete programming models (including nondeterminism, fairness, and probabilistic choice)

2. D. Beyer, N.-Z. Lee: The Transformation Game: Joining Forces for Verification, Springer LNCS 15262, 2025
   - survey of verification-oriented, modular language transformations and their applications



D. Static Analysis of Quantum Programs

1. P. Zhao, X. Wu, Z. Li, J. Zhao: QChecker: Detecting Bugs in Quantum Programs via Static Analysis, Q-SE 2023
   - presents a static analysis tool for finding bugs in quantum programs in Qiskit (incorrect use of quantum gates, measurement-related issues, incorrect initial states, ...)
2. M. Paltenghi, M. Pradel: Analyzing Quantum Programs with LintQ: A Static Analysis Framework for Qiskit, FSE 2024
   - another static analysis framework for detecting bugs in quantum programs (corrupted quantum states, redundant measurements, incorrect compositions of sub-circuits, ...)
3. S. Xia, J. Zhao: Static Entanglement Analysis of Quantum Programs, Q-SE 2023
   - static code analysis technique to determine which qubit may entangle with another qubit, resulting in entanglement graph

```
simulator = Aer.get_backend("qasm_simulator")

qreg = QuantumRegister(3)
creg = ClassicalRegister(3)
circuit = QuantumCircuit(qreg, creg)

circuit.h(0)
circuit.h(2)
circuit.cx(0, 1)
circuit.measure([0,1,2], [0,1,2])
job = execute(circuit, simulator, shots=1000)
result = job.result()
counts = result.get_counts(circuit)
print(counts)
```

Outline

Overview

Aims of this Seminar

Important Dates

The Topics

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!