



Seminar *Trends in Computer-Aided Verification*

Introduction

Winter 2026/27; October 13, 2026

Thomas Noll, Xaver Fink, Philipp Schroer
Software Modeling and Verification Group
RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ws-2026-27/cav/>

Outline

Overview

Aims of this Seminar

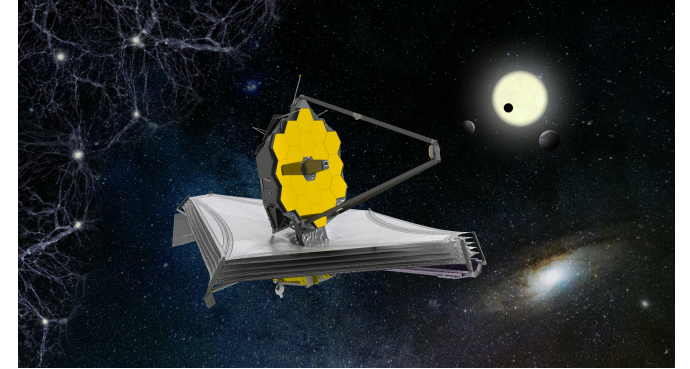
Important Dates

A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

C. Analysing Quantum Software [Thomas Noll]

Final Hints

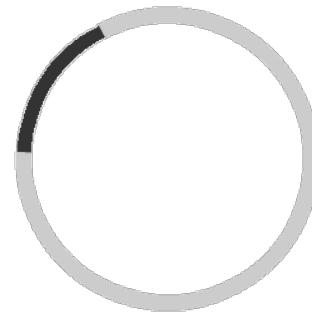
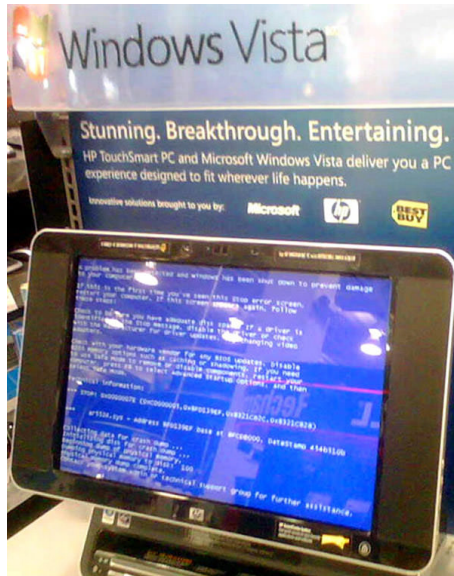


Software Systems are everywhere!





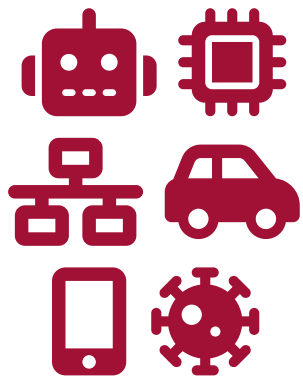
Software **Failures** are everywhere!



Loading..



System Model



System Property



+ diagnostics

Outline

Overview

Aims of this Seminar

Important Dates

A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

C. Analysing Quantum Software [Thomas Noll]

Final Hints

Objectives of this Seminar

Objectives

- Independent understanding of a scientific research article
- Describe the problem(s) considered in the article and the necessary background
- Explain relevant research results at an adequate level of detail
- Acquire, read, and understand related literature
- Write a concise report and give an oral presentation covering the above points

Requirements on Report

Your report

- Independent writing of a **concise report of two pages**
 - Can use the $\text{\LaTeX}$ template provided on the website
 - Must be understandable to everyone in this room
- **First milestone:** detailed outline & 20% of technical content written
 - Get a deep understanding of your topic — we will ask you about it!
 - Provide overview of structure (section headers, main definitions/theorems)
- Correct **spelling** and **grammar** is required; use German or English
 - We abort correction if there are too many errors
- Wise use of **generative AI**
 - e.g. just for grammar correction
- **Second milestone:** 100% done
 - **Complete** and **correctly cited** set of references to all consulted literature
 - Additional content can be put into an **appendix** (optional)
 - References and appendices do not count for the page limit
- **Third milestone:** submission of final (“camera-ready”) version

Requirements on Presentation

Your presentation

- **Total duration:** 30 minutes (20 minutes presentation time + 10 minutes for Q&A)
- Finish in time – overtime is bad
- Focus your talk on the **audience**, abstract away from details as necessary
- **Descriptive slides** (L^AT_EX template on website, can also use other software)
- ≤ 15 lines of text per slide, use colors, number your slides, correct spelling
- Prepare for **expected questions**, e.g., with backup slides

The presentation and Q&A make up at least half of the grade!

Outline

Overview

Aims of this Seminar

Important Dates

A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

C. Analysing Quantum Software [Thomas Noll]

Final Hints

Important Dates

Deadlines

- October 15: Topic preferences due
- November 16: Detailed outline with 20% content due
- December 7: Full report due
- January 7: Presentation slides due
- January 18–21 (?): Seminar talks

Important

- Missing a deadline causes **immediate exclusion** from the seminar.
- 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) tries by one.

Selecting Your Topic

Procedure

- Enter the poll in the link you will receive via e-mail after this event
 - <https://terminplaner.dfn.de/public/bookings/08f92d33-608e-4a20-8d6e-5a1333adac12>
 - Give preferences and/or nickname, if applicable
 - Do this until **Thursday, October 15**
- We do our best to find a good topic-student assignment
 - It helps to indicate several topics
- Assignment will be announced on Friday, October 16



Outline

Overview

Aims of this Seminar

Important Dates

A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

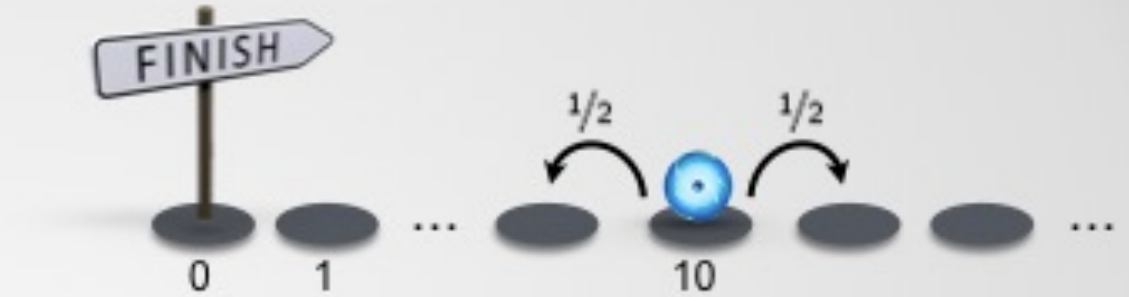
C. Analysing Quantum Software [Thomas Noll]

Final Hints

Probabilistic Programming

A particle starts at position $x=10$ and moves with equal probability to the left or to the right in each turn. The random walk stops when the particle reaches position $x=0$.

What is the expected number of moves to termination?



```

$$C_{rw}: \quad x := 10; \text{ while } (x > 0) \{ x := \frac{1}{2} \cdot \langle x-1 \rangle + \frac{1}{2} \cdot \langle x+1 \rangle \}$$

```

Differential Privacy via Probabilistic Verification

- Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Đorđe Žikelić: *SuperDP: Differential Privacy Refutation via Supermartingales*, PLDI 2026
- Differential privacy: formal concept that tackles the problem of extracting statistical information about a dataset without compromising the privacy of individual data subjects.
- Paper about refuting ϵ -differential privacy using **expectation witnesses** and super-/submartingales.

Lower Bounds for Probabilistic Programs

- Satoshi Kura, Hiroshi Unno, Takeshi Tsukada *Supermartingales for Unique Fixed Points: A Unified Approach to Lower Bound Verification*, PLDI 2026
- Many quantitative properties of probabilistic programs can be characterized as least fixed points.
- Paper uses generalized ranking supermartingales to establish **unique fixed points**, yielding lower bounds for termination probabilities, expected runtime, etc.

Understanding SMT Solver Behaviour

- Jonáš Fiala, Peter Müller: *SMTScope: Automated and Efficient Analysis of SMT Traces*, TACAS 2026
- SMT solvers (such as Z3) enable the automated verification of complex software, but also cause performance problems, proof brittleness, and spurious errors.
- Paper analyses and visualizes Z3 traces to diagnose **quantifier instantiations, matching loops, and lurking axioms**.

Outline

Overview

Aims of this Seminar

Important Dates

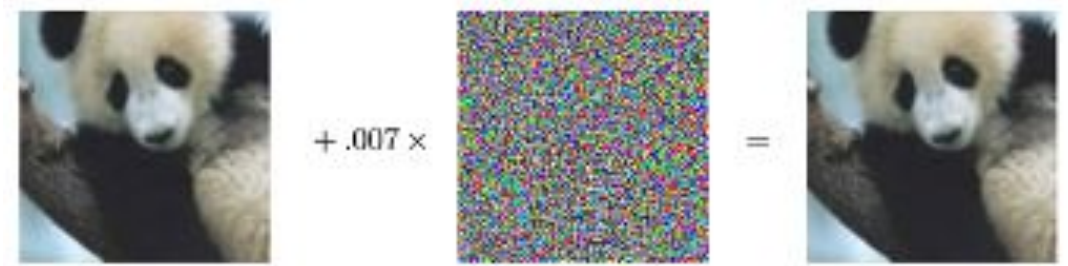
A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

C. Analysing Quantum Software [Thomas Noll]

Final Hints

Motivation



- **Verification** guarantees robustness to perturbations
 - Formal process, sound bounds on network behaviour
- **Novelty Detection** identifies unexpected inputs
 - Heuristic approach
 - Aims to avoid “guessing” for inputs the network was not trained on

Reusing Knowledge in Neural Network Verification

- Raya Elsaleh, Liam Davis, Haoze Wu, Guy Katz: [Incremental Neural Network Verification via Learned Conflicts](#), FMCAD 2026
- Can information from one neural network verification query be reused for solving other closely related queries?
- The paper shows a way to do this using learned conflict clauses and proves soundness of the application of this to branch-and-bound in neural network verification by constructing refinement relations.

Equivalence Verification of Neural Networks

- Samuel Teuber, Philipp Kern, Marvin Janzen, Bernhard Beckert: [Revisiting Differential Verification: Equivalence Verification with Confidence](#), TACAS 2025
- Can we prove that a modified neural network (e.g., after pruning or retraining) still behaves equivalently to the original neural network?
- The paper uses differential reasoning over a novel abstract domain to reason directly about differences between the two networks and investigates which notions of equivalence can be verified efficiently.

Outline

Overview

Aims of this Seminar

Important Dates

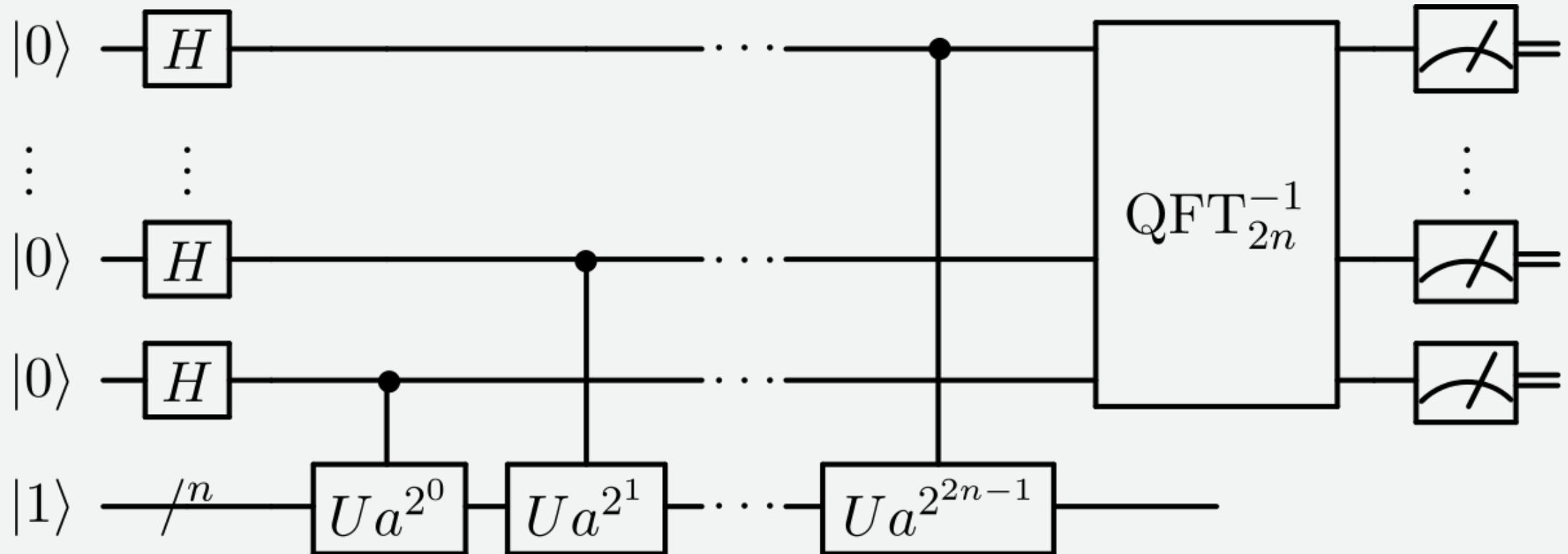
A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

C. Analysing Quantum Software [Thomas Noll]

Final Hints

A Quantum Program



1. Detecting Quantum Bugs using LintQ

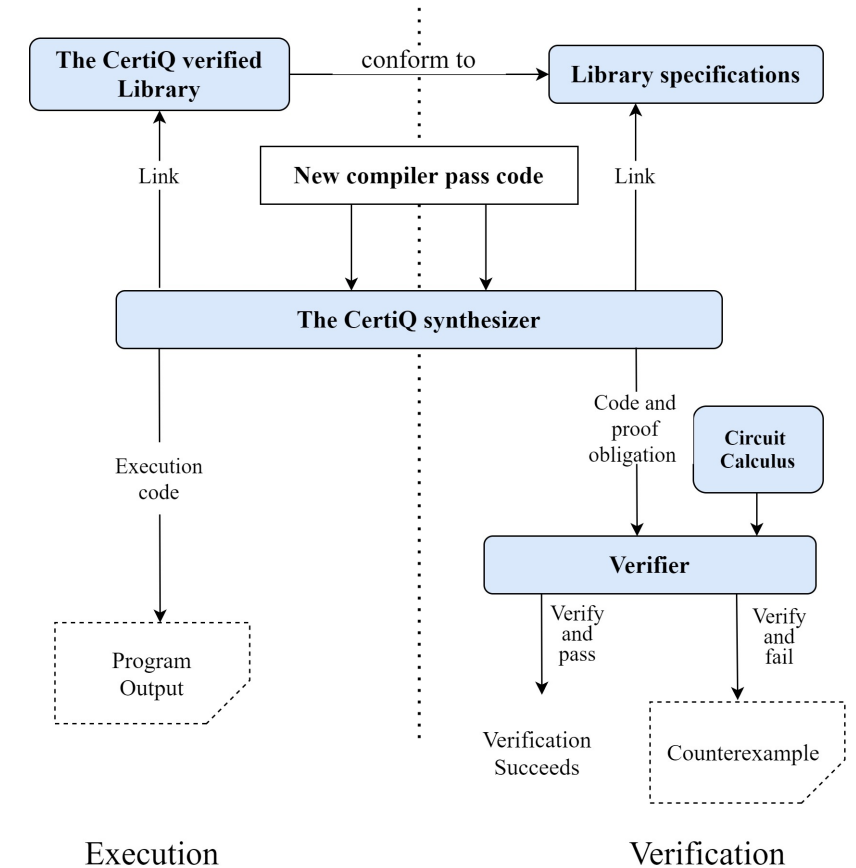
```
1 qc = QuantumCircuit(2, 2)
2 qc.h(1)
3 qc.cx(1, 0)
4 qc.measure(0, 0)
5 qc.measure(1, 1)
6 qc.z(0) # Problem: Qubit 0 has collapsed
7 qc.measure(0, 0)
```

```
1 from Measurement m, Gate g, int q
2 where
3     mayFollowDirectly(m, g, q)
4     and not g.isConditional()
5 select gate, "Gate after measurement
   on qubit " + q
```

- Matteo Paltenghi, Michael Pradel: *Analyzing Quantum Programs with LintQ: A Static Analysis Framework for Qiskit*, FSE 2024
- Uses **abstractions** for reasoning about common concepts in quantum computing (without referring to details of underlying quantum computing platform)
- Offers an **extensible set of ten analyses** that detect likely bugs
 - operating on corrupted quantum states, redundant measurements, incorrect compositions of sub-circuits, ...

2. Verifying Quantum Compilers using CertiQ

- Yunong Shi et al.: *CertiQ: Mostly-automated Verification of a Realistic Quantum Compiler*, arXiv:1908.08963
- Framework for writing and verifying compiler passes of Qiskit
- Generates verification conditions from source-code annotations



Outline

Overview

Aims of this Seminar

Important Dates

A. Probabilistic Programming [Philipp Schroer]

B. Verification of Neural Networks [Xaver Fink – remote!]

C. Analysing Quantum Software [Thomas Noll]

Final Hints

Some Final Hints

- Get an understanding of the **practical and theoretical (!) aspects** of the publication
 - **Reminder:** this is a seminar in theoretical computer science
- Find the right **level of detail**
 - Your report and talk should be **self-contained and understandable** to people that are not familiar with the original work
 - Focus on **core ideas**, omit too specific details (e.g., related work or optimisations)
 - Provide **examples**
- Discuss contents / ideas / problems with your **supervisor**
 - Contact them on time, prepare the meetings
- **Take your time**
 - Seminar yields 4 credit points
 - Officially, this translates to around $4 \times 30 = 120$ hours of work
 - Do not expect to write the report / prepare the talk in a single day ...