Programmierparadigmen

Prof. Dr. Sebastian Erdweg
Programming Languages, PL@KIT

Programming Languages, PL@KIT

Research Topics

- Language design and DSLs
- Program analysis and type systems
- Incremental computing
- Datalog expressivity and execution
- Metaprogramming

New courses

- WS 25/26: Seminar “Type systems”
- SS 26/27: Course “Program analysis” (planned)
- SS 26/27: Practical “Interactive theorem proving” (planned)

Schedule — Lectures, Exercises, and Exam

Face-to-face meetings

- Wednesday, 14:00 – 15:30 Uhr
  Hertz Lecture Hall (Building 10.11, room 126)
- Friday, 14:00 – 15:30 Uhr
  Hertz Lecture Hall (Building 10.11, room 126)

The distribution of lectures and exercises varies — see ILIAS for details.

Exam

- Friday, 20 March 2026, 08:00 – 10:00
- No open-book exam
- Formal prerequisite: Modul Theoretische Grundlagen der Informatik

Lehramt: Probably won’t need Programmierparadigmen anymore.
You get an email this week.

Exercises

Weekly tasks, published on ILIAS

- Working through the worksheets is very strongly recommended. The exercises are the best preparation for the final exam.
- No tutorials, classroom session (Saalübung).
- Come prepared to the classroom session.
- Ask questions in the classroom and in the ILIAS forum.

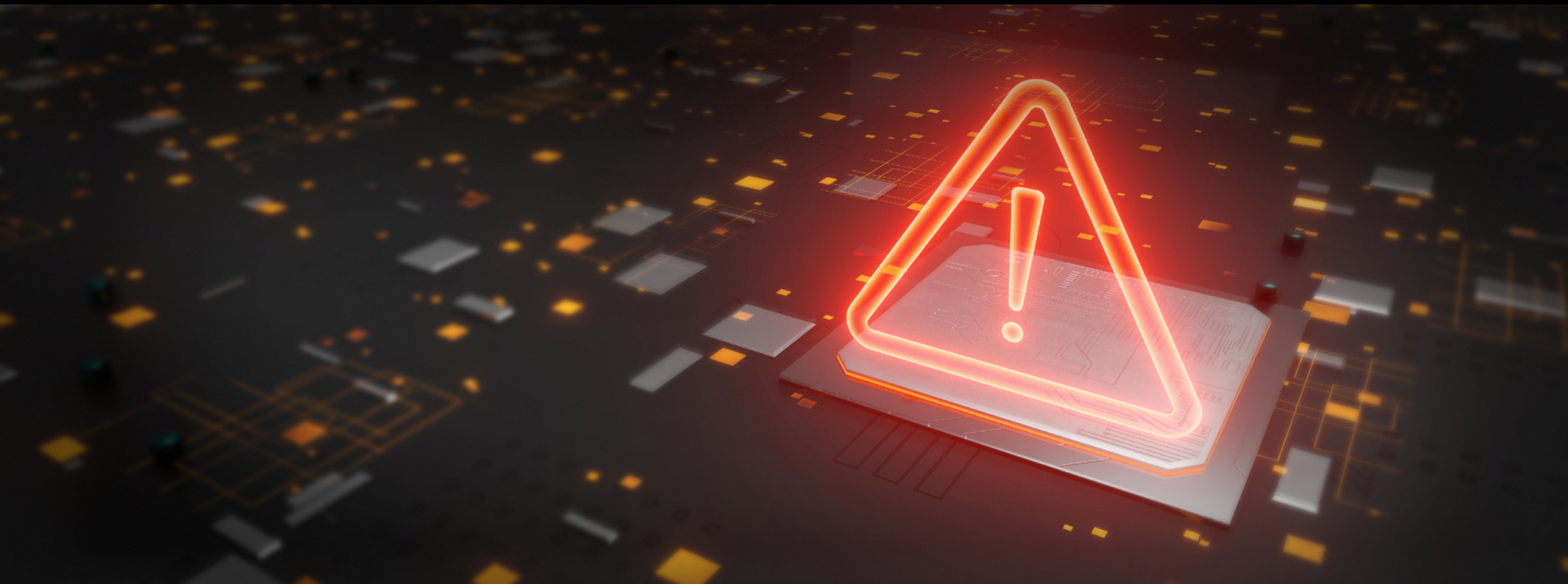
Mo Bitar

Runqing Xu

https://ilias.studium.kit.edu/ilias.php?baseClass=ilrepositorygui&ref_id=2757025

pp-vorlesung@cs.kit.edu

This course has new content!



Functional Programming Techniques

— What you need to know to develop software using functional programming —

Functional programming (FP)

- Data modeling and processing
- Recursive data and recursive functions
- Complexity and performance
- Specification and validation
- Functional programming with effects

Parallel programming by Prof. Dr. Ralf Reussner

Week 1–5: Functional programming
Week 6–9: Parallel programming
Week 10–14: Functional programming

Why learn functional programming?

- **Immutable data reduces bugs** by eliminating unintended state changes.
- **Pure functions simplify reasoning** because outputs depend only on inputs.
- No shared state **makes concurrency safer** and easier to implement.
- Function composition allows **building complex logic from simple parts**.
- **Industry demand is growing**: FP is common in data engineering, distributed systems, and functional features in mainstream languages (e.g., Java, Python, JavaScript, Scala).

Teaching style — Live programming in Lean

- Live development of the material in front of the class.
- The commented source code serves as lecture notes.
- We exclusively use Lean for functional programming in this course.

Why use Lean?

- Purely functional language – enforces FP principles by design.
- Strong type system – demonstrates advanced FP concepts.
- Mathematical rigor – links FP to formal logic and proofs.
- Interactive tooling – easy experimentation with FP ideas.

How to get started with Lean?
=> **Classroom session this Friday**

Literature on Functional Programming

- FP basics in Lean
  “Functional Programming in Lean”
  https://lean-lang.org/functional_programming_in_lean/

- Advanced Lean and verification
  "Hitchhiker's Guide to Logical Verification”
  https://github.com/lean-forward/logical_verification_2025

- Reference books about FP theory
  “Types and Programming Languages” by Benjamin C. Pierce
  “Practical Foundations for Programming Languages” by Robert Harper

Exam retakes

- If this is your **first attempt** at Programmierparadigmen,
  your upcoming exam will by default be about the **new topics**.

- If you **previously failed** the exam of Programmierparadigmen,
  your upcoming retake exam will by default be about the **previous topics**.
- You can opt-in to the new topics via the ISS (details to follow).

- It will be visible to you soon whether you are register for the new or old topics.
- More information to follow, please refrain from contacting us about this now.


LET'S GET
STARTED