RIO 2025 Summer School: Isabelle Course

Peter Lammich

University of Twente

March 2025 @ Rio Cuarto

About the Instructor

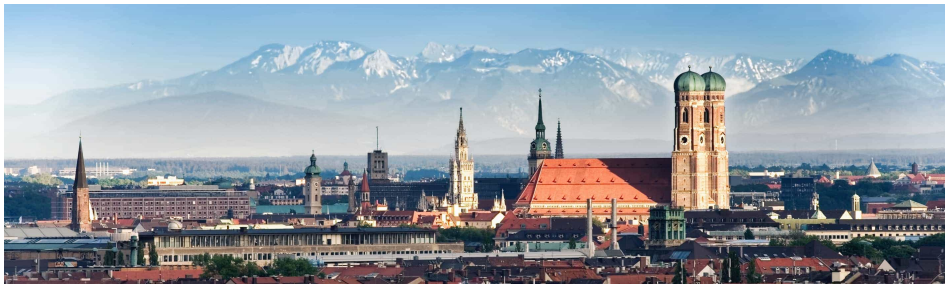
Peter Lammich, using Isabelle since 20 years

About the Instructor

Peter Lammich, using Isabelle since 20 years
Made my PhD in Münster, Germany

About the Instructor

Peter Lammich, using Isabelle since 20 years
Made my PhD in Münster, Germany
Then Munich



About the Instructor

Peter Lammich, using Isabelle since 20 years
Made my PhD in Münster, Germany
Then Munich, VirginiaTech

About the Instructor

Peter Lammich, using Isabelle since 20 years
Made my PhD in Münster, Germany
Then Munich, VirginiaTech, Manchester

About the Instructor

Peter Lammich, using Isabelle since 20 years
Made my PhD in Münster, Germany
Then Munich, VirginiaTech, Manchester
At UTwente since March 2021

Email p.lammich@utwente.nl

About the Course

- Goals
  - you'll be able to use Isabelle/HOL
  - verify simple functional programs
  - learn the basics to dive deeper
- 4 lessons, 2:30h each
- Starts with lecture, then hands-on exercises
- Bram (and maybe Edo) will help you
- Books (freely available online)
  - Nipkow, T. (Ed). (2024). *Functional Data Structures and Algorithms: A Proof Assistant Approach* https://functional-algorithms-verified.org/
  - Nipkow, T. and Klein, G. (2014). *Concrete Semantics*. Springer. https://www.concrete-semantics.org/
  - Nipkow, T. *Programming and Proving in Isabelle/HOL* https://isabelle.in.tum.de/dist/Isabelle2024/doc/prog-prove.pdf

Software Errors

Software is buggy!

Software Errors

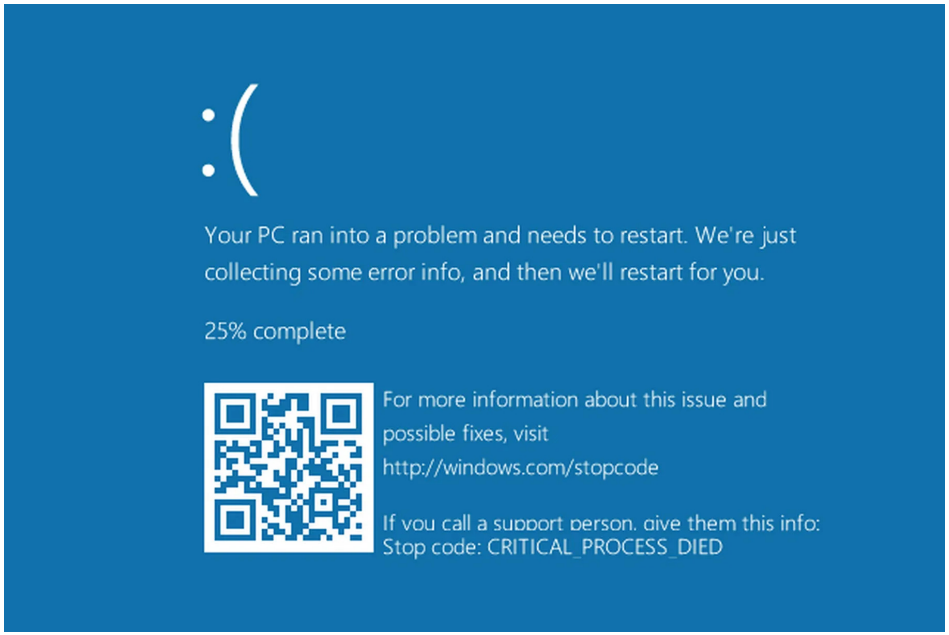
Software is buggy!

Consequences:

Software Errors

Software is buggy!

Consequences:annoying



Software Errors

Software is buggy!

Consequences:annoying, expensive

Software Errors

Software is buggy!

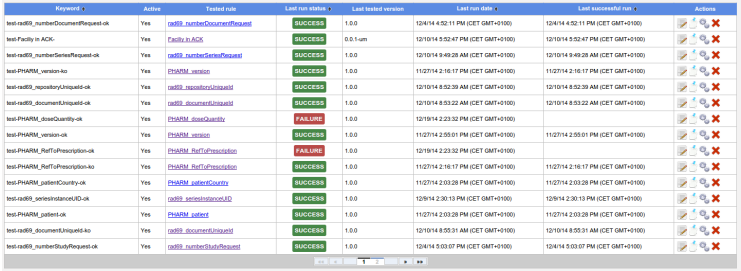
Consequences:annoying, expensive, **deadly**!

Software Verification

Testing?

Software Verification

Testing? Won't find all bugs!



Keyword	Active	Tested rule	Last run status	Last tested version	Last run date	Last successful run	Actions
test-rad69_numberDocumentRequest-ok	Yes	rad69_numberDocumentRequest	SUCCESS	1.0.0	12/4/14 4:52:11 PM (CET GMT+0100)	12/4/14 4:52:11 PM (CET GMT+0100)	
test-Facility in ACK-	Yes	Facility in ACK	SUCCESS	0.0.1-um	12/10/14 5:52:47 PM (CET GMT+0100)	12/10/14 5:52:47 PM (CET GMT+0100)	
test-rad69_numberSeriesRequest-ok	Yes	rad69_numberSeriesRequest	SUCCESS	1.0.0	12/10/14 9:49:28 AM (CET GMT+0100)	12/10/14 9:49:28 AM (CET GMT+0100)	
test-PHARM_version-ko	Yes	PHARM_version	SUCCESS	1.0.0	11/27/14 2:16:17 PM (CET GMT+0100)	11/27/14 2:16:17 PM (CET GMT+0100)	
test-rad69_repositoryUniqueId-ok	Yes	rad69_repositoryUniqueId	SUCCESS	1.0.0	12/10/14 8:52:39 AM (CET GMT+0100)	12/10/14 8:52:39 AM (CET GMT+0100)	
test-rad69_documentUniqueId-ok	Yes	rad69_documentUniqueId	SUCCESS	1.0.0	12/10/14 8:53:22 AM (CET GMT+0100)	12/10/14 8:53:22 AM (CET GMT+0100)	
test-PHARM_doseQuantity-ok	Yes	PHARM_doseQuantity	FAILURE	1.0.0	12/19/14 2:23:32 PM (CET GMT+0100)		
test-PHARM_version-ok	Yes	PHARM_version	SUCCESS	1.0.0	11/27/14 2:55:01 PM (CET GMT+0100)	11/27/14 2:55:01 PM (CET GMT+0100)	
test-PHARM_RefToPrescription-ok	Yes	PHARM_RefToPrescription	FAILURE	1.0.0	12/19/14 2:23:32 PM (CET GMT+0100)		
test-PHARM_RefToPrescription-ko	Yes	PHARM_RefToPrescription	SUCCESS	1.0.0	11/27/14 2:16:17 PM (CET GMT+0100)	11/27/14 2:16:17 PM (CET GMT+0100)	
test-PHARM_patientCountry-ok	Yes	PHARM_patientCountry	SUCCESS	1.0.0	11/27/14 2:03:28 PM (CET GMT+0100)	11/27/14 2:03:28 PM (CET GMT+0100)	
test-rad69_seriesInstanceUID-ok	Yes	rad69_seriesInstanceUID	SUCCESS	1.0.0	12/9/14 2:30:13 PM (CET GMT+0100)	12/9/14 2:30:13 PM (CET GMT+0100)	
test-PHARM_patient-ok	Yes	PHARM_patient	SUCCESS	1.0.0	11/27/14 2:03:28 PM (CET GMT+0100)	11/27/14 2:03:28 PM (CET GMT+0100)	
test-rad69_documentUniqueId-ko	Yes	rad69_documentUniqueId	SUCCESS	1.0.0	12/10/14 8:55:31 AM (CET GMT+0100)	12/10/14 8:55:31 AM (CET GMT+0100)	
test-rad69_numberStudyRequest-ok	Yes	rad69_numberStudyRequest	SUCCESS	1.0.0	12/4/14 5:03:07 PM (CET GMT+0100)	12/4/14 5:03:07 PM (CET GMT+0100)	

Software Verification

Testing? Won't find all bugs!

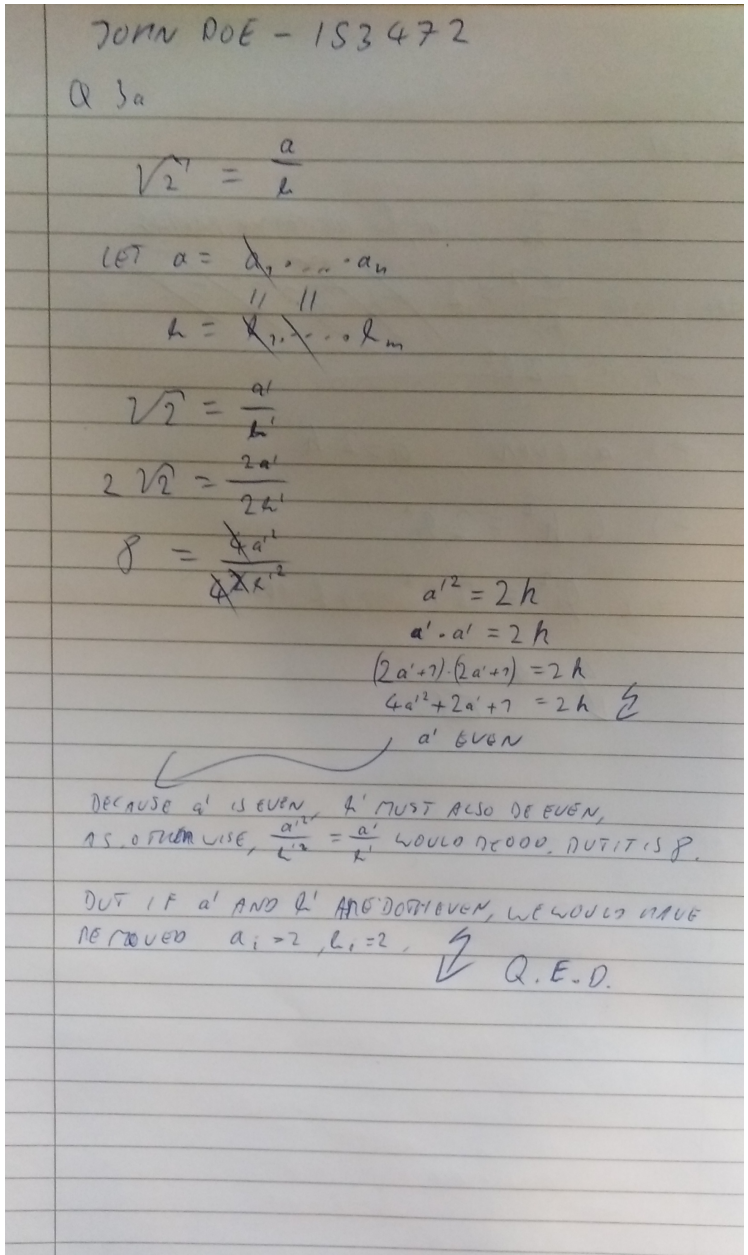
Mathematical proof that SW is correct?

Q.E.D.

Software Verification

Testing? Won't find all bugs!

Mathematical proof that SW is correct?
Proofs (pen&paper) are error prone!



Software Verification

Testing? Won't find all bugs!

Mathematical proof that SW is correct?
Proofs (pen&paper) are error prone!
In particular those about software!

Line number	Formal proof	Justification
1	$\mathbf{true} \supset x = x + y \times 0$	Lemma 1
2	$x = x + y \times 0\{r := x\}x = r + y \times 0$	D0
3	$x = r + y \times 0\ \{q := 0\}\ x = r + y \times q$	D0
4	$\mathbf{true}\ \{r := x\}\ x = r + y \times 0$	D1 (1, 2)
5	$\mathbf{true}\ \{r := x;\ \ q := 0\}\ x = r + y \times q$	D2 (4, 3)
6	$x = r + y \times q \wedge y \leqslant r \supset x = (r-y) + y \times (1+q)$	Lemma 2
7	$x = (r-y) + y \times (1+q)\{r := r-y\}x = r + y \times (1+q)$	D0
8	$x = r + y \times (1+q)\{q := 1+q\}x = r + y \times q$	D0
9	$x = (r-y) + y \times (1+q)\{r := r-y;\ q := 1+q\}\ x = r + y \times q$	D2 (7, 8)
10	$x = r + y \times q \wedge y \leqslant r\ \{r := r-y;\ q := 1+q\}\ x = r + y \times q$	D1 (6, 9)
11	$x = r + y \times q\ \{\mathbf{while}\ y \leqslant r\ \mathbf{do}\ (r := r-y;\ \ q := 1+q)\}$ $\neg y \leqslant r \wedge x = r + y \times q$	D3 (10)
12	$\mathbf{true}\ \{((r := x;\ \ q := 0);\ \ \mathbf{while}\ y \leqslant r\ \mathbf{do}\ (r := r-y;\ \ q := 1+q))\}\ \neg y \leqslant r \wedge x = r + y \times q$	D2 (5, 11)

Software Verification

Testing? Won't find all bugs!

Mathematical proof that SW is correct?
Proofs (pen&paper) are error prone!
In particular those about software!

Let computer check the proof?

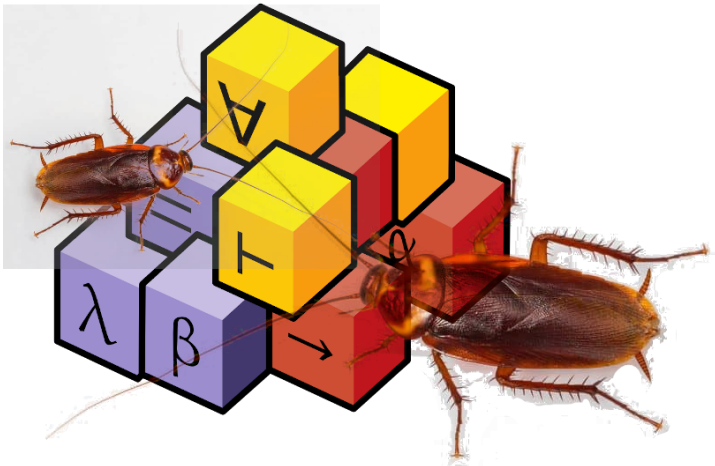
```
theorem sqrt2_not_rational:
  "sqrt 2 ∉ ℚ"
proof
  let ?x = "sqrt 2"
  assume "?x ∈ ℚ"
  then obtain m n :: nat where
    sqrt_rat: "¦?x¦ = m / n" and lowest_terms: "coprime m n"
    by (rule Rats_abs_nat_div_natE)
  hence "m^2 = ?x^2 * n^2" by (auto simp add: power2_eq_square)
  hence eq: "m^2 = 2 * n^2" using of_nat_eq_iff power2_eq_square by fastforce
  hence "2 dvd m^2" by simp
  hence "2 dvd m" by simp
  have "2 dvd n" proof -
    from ‹2 dvd m› obtain k where "m = 2 * k" ..
    with eq have "2 * n^2 = 2^2 * k^2" by simp
    hence "2 dvd n^2" by simp
    thus "2 dvd n" by simp
  qed
  with ‹2 dvd m› have "2 dvd gcd m n" by (rule gcd_greatest)
  with lowest_terms have "2 dvd 1" by simp
  thus False using odd_one by blast
qed
```

Software Verification

Testing? Won't find all bugs!

Mathematical proof that SW is correct?
Proofs (pen&paper) are error prone!
In particular those about software!

Let computer check the proof?
But you have to trust proof checker!

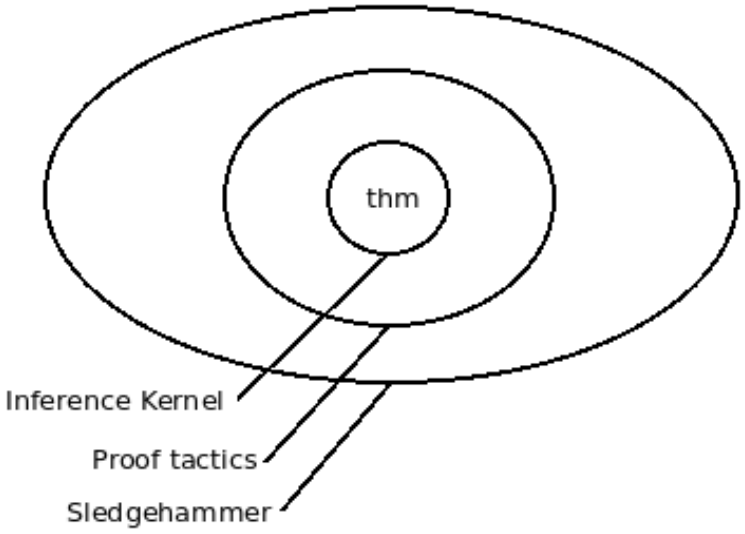


Software Verification

Testing? Won't find all bugs!

Mathematical proof that SW is correct?
Proofs (pen&paper) are error prone!
In particular those about software!

Let computer check the proof?
But you have to trust proof checker!
Can be designed with small TCB!



Isabelle/HOL Interactive Theorem Prover

Isabelle/HOL Interactive Theorem Prover

- based on Higher-Order Logic

Isabelle/HOL Interactive Theorem Prover

- based on Higher-Order Logic
- trustworthy: small inference kernel

Isabelle/HOL Interactive Theorem Prover

- based on Higher-Order Logic
- trustworthy: small inference kernel
- powerful automation (e.g. sledgehammer)

Isabelle/HOL Interactive Theorem Prover

- based on Higher-Order Logic
- trustworthy: small inference kernel
- powerful automation (e.g. sledgehammer)
- large collection of libraries

Isabelle/HOL Interactive Theorem Prover

- based on Higher-Order Logic
- trustworthy: small inference kernel
- powerful automation (e.g. sledgehammer)
- large collection of libraries
- Archive of Formal Proofs

Isabelle/HOL Interactive Theorem Prover

- based on Higher-Order Logic
- trustworthy: small inference kernel
- powerful automation (e.g. sledgehammer)
- large collection of libraries
- Archive of Formal Proofs
- mature, production quality IDE, based on JEdit

What can be done with Isabelle

- Verifying operating systems
  - seL4 https://sel4.systems/
- Proving sophisticated maths
  - Kepler conjecture (Hales' thm) https://github.com/flyspeck/flyspeck
- And many more. See https://www.isa-afp.org/

What I do:

- Verifying algorithms and software
  - From specification down to efficient LLVM IR
  - (Parallel) sorting algorithms
  - Model-Checkers
  - SAT-solvers
  - Numeric algorithms

What we'll do in this course

- Hands-on basics of Isabelle
- Verify algorithms on natural numbers and lists
  - down to (more or less efficient) Haskell
- Give you the basics to dive deeper