



# Program Verification

## Part 8 – Summary and Outlook

René Thiemann

Department of Computer Science

## Summary of Course

## Summary by Parts

- part 2: extend **first-order logic** (of Logic course) by **types**
- part 3: define **standard model** for **well-defined functional programs**;
- part 4: methods to ensure **well-definedness** of functional programs, including automated **termination analysis**
- part 5: derive **axioms** for induction, equality of constructors, etc.; framework for **induction proofs and equational reasoning**; **specifications can be given via functional programs**
- part 6: verification of **imperative programs** via **Hoare-calculus**; includes formal **semantics** and proof of **soundness of calculus verification condition generation**;
- part 7: **certification**  
main software stays unverified; generate justifications for each output; verified certifier checks these justifications

## Summary by Methodology

- inductively defined sets
- proofs by induction in various settings  
(by algorithm, by data-structure, by inductively defined set, ...)
- proofs by invariants
- verification by refinement
  - prove soundness of (abstract) pseudo-code against specification
  - prove that concrete code is valid implementation pseudo-code
- integrating external tools and certification  
termination proofs via SMT-solver, logic-solver for Hoare-calculus
- development of paper-verified interpreter for functional programs written in Haskell
  - checks well-definedness of input (missing: termination analyser)
  - algorithms for these checks have been verified
  - verified implementation of one-step evaluation  $\leftrightarrow$

## Feedback

- feedback is highly welcome  
(via mail, anonymous via PV-website, via evaluation, etc.)
  - content + structure
  - feasibility
  - typos
  - ...

# Outlook

## Related Courses

- backend-solvers: constraint solving, automated theorem proving
- core evaluation mechanism: (selected topics in) term rewriting
- program verification with tool support: interactive theorem proving
- more automation: program analysis

## Related Bachelor Thesis Topics

- recently finished
  - efficient implementation of weighted path order (verified)
  - translation of multitape Turing machines into singletape TMs (partly verified)
  - decision procedure for termination of right-ground term rewrite systems (verified)
  - encoding of multiset-comparisons into SAT (verified)
- ongoing
  - two projects related to FP and to SAT solving (no verification aspect)
  - automation of rewriting induction with machine checkable proof generation (certification)
- available
  - optimizing an algorithm for multiset-comparisons (verified)
  - Certification Problem Format Visualizer (certification)
  - always: contact me with your own ideas on program verification related topics



**Thank you for your interest!**