Proving with Computer Assistance
Lecture 1

Herman Geuvers

Administration

- Teacher: Herman Geuvers (Thursday only)
- Mail to `herman@cs.ru.nl`
- Web page:
  `http://www.cs.ru.nl/H.Geuvers/onderwijs/provingwithCA/`
  or look at the link at my homepage.
- Weekly overview: see the webpage
- Lectures will not be recorded, but recordings of last year's lectures are on Canvas.
- For practical work we will use the Proof Assistant Coq, which you can install yourself. See the webpage.
- We will be working on Coq .v files that will be provided via the webpage.

Examination

- Written exam + Coq Assignment
- Final grade = (Written Exam + Coq Assignment)/2
  with the condition that your Written Exam mark should be 5 or higher.
- If your Written Exam mark is below 5, this is your Final grade.
- You don't receive a mark (so I will write "NV") if you haven't completed all parts of the course.
- Written exam: Monday April 15, Time: 13:30–16:30.
  It is an *open book exam*, so you can bring any paper material you want
- Deadline Coq Assignment: Wednesday April 17.
- In the resit period you can "redo" the written exam and/or the assignment. Marks from the first period will be retained.

Content

- Logic, Natural Deduction (known?)
- Lambda calculus (known?)
- Working with the Proof Assistant Coq
- Type Theory

The general picture

What are Proof Assistants for?

- Precise mathematical modelling (defining)
- Verification of properties of systems (proving)

Computer supports in these activities:

- Checking correctness of definitions
- Take care of the bookkeeping
- Do some computation
- Do some proving for us

The general picture

Does the Proof Assistant do all the proving for us?

No . . .

It is undecidable in general whether a formula is true or not.

Automated Theorem Provers	Proof Assistants
Specific domains	Generally applicable
Massage your problem	Modelling is direct
False or True (or Don't Know)	Interactive, user guided
No proofs	complete, checkable proofs

The general picture

- Automated Theorem Provers
  E.g. Vampire, CVC5, Z3, Otter, ACL2 . . . Specialized (e.g. logic programs, satisfiability problems), Built-in automation (e.g. resolution, SMT)
- Model Checkers
  E.g. MCRL2, Uppaal, Spin, SMV, . . . Specialized (reachability problems), Built-in automation (state space abstraction)
- Computer Algebra Systems
  E.g. Maple, Mathematica . . . Specialized (solving equations over $\mathbb{C}$), Built-in automation (symbolic term rewriting), may give wrong answer.
- Proof Assistants
  E.g. Coq, Lean, Isabelle, PVS, HOL, Agda, Mizar, . . . Generic, Little automation (program your own . . .)

Use of PAs

Who is using Proof Assistants and what for?
Computer Scientists for

- Modelling and specifying systems
- Proving the correctness of models / software /systems

Mathematicians for

- Building up theories
- Verifying proofs

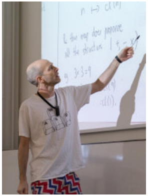
Mathematicians are not (yet) big users of Proof Assistants

- Mechanically verifying a proof takes too much time. (Too much idiosyncracy, not enough automation.)
- We don't need computers to verify proofs! We are much better at it!

Mathematical users of Proof Assistants

Gradually, more mathematicians are getting interested, young mathematicians are less afraid of computers.

- Store formalized mathematics on a computer and make large repositories of formal mathematics actively available.
- Various mathematicians observe that the proofs in their field are becoming too long, complex, abstract that one can only trust them if they are machine verified.
- Kevin Buzzard: Mathlib
  a user maintained library for the
  Lean theorem prover



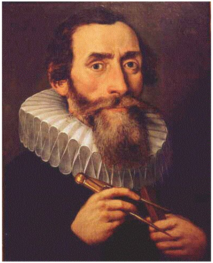
Large example of a mathematical use of Proof Assistants

Flyspeck project: Formalizing a proof of the Kepler Conjecture

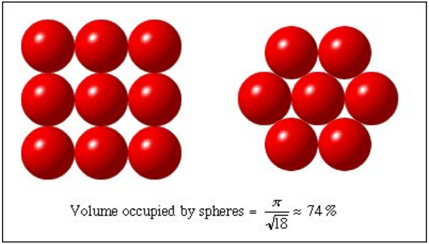
`http://code.google.com/p/flyspeck/`

Tom Hales, CMU Pittsburgh

Kepler Conjecture (1611)



The most compact way of stacking balls of the same size is a pyramid.



Kepler Conjecture (1611)

- Hales 1998: proof of the conjecture using computer programs (article of 300 pages)

Thomas Hales, associate professor of mathematics, demonstrates his solution to the Kepler conjecture, a problem that mathematicians have been wrestling with since 1611. Tennis balls courtesy of the Varsity Tennis Club. Photo by Bob Kalmbach

- Annals of Mathematics: reviewers state it is 99% correct
  ...but we can't verify the correctness of the computer programs.

Hales' proof of the Kepler conjecture

Reduce the problem to 1039 inequalities of the shape

$$\frac{\begin{matrix}-x_1x_3 - x_2x_4 + x_1x_5 + x_3x_6 - x_5x_6 + \\ x_2(-x_2 + x_1 + x_3 - x_4 + x_5 + x_6)\end{matrix}}{\sqrt{4x_2\begin{pmatrix} x_2x_4(-x_2 + x_1 + x_3 - x_4 + x_5 + x_6) + \\ x_1x_5(x_2 - x_1 + x_3 + x_4 - x_5 + x_6) + \\ x_3x_6(x_2 + x_1 - x_3 + x_4 + x_5 - x_6) \\ -x_1x_3x_4 - x_2x_3x_5 - x_2x_1x_6 - x_4x_5x_6 \end{pmatrix}}} < \tan(\frac{\pi}{2} - 0.74)$$

Use computer programs to verify these inequalities.

Flyspeck project

- Hales: formalise the proof of Kepler's conjecture using Proof Assistants Write the computer code in the PA, prove it correct in the PA and run it in the PA.
- Proof Assistants used: Hol-light, Isabelle, Coq

Computer Science users of Proof Assistants

Compcert (Leroy et al.)

- verifying an optimizing compiler from C to x86/ARM/PowerPC code
- implemented using Coq's functional language
- verified using using Coq's proof language

Xavier Leroy

why?

- your high level program may be correct, maybe you've proved it correct ...
- ... but what if it is compiled to wrong code?
- compilers do a lot of optimizations: switch instructions, remove dead code, re-arrange loops, ...
- for critical software the possibility of miscompilation is an issue

C-compilers are generally not correct

Csmith project *Finding and Understanding Bugs in C Compilers*, X. Yang, Y. Chen, E. Eide, J. Regehr, University of Utah.

> *... we have found and reported more than 325 bugs in mainstream C compilers including GCC, LLVM, and commercial tools.*
> *Every compiler that we have tested, including several that are routinely used to compile safety-critical embedded systems, has been crashed and also shown to silently miscompile valid inputs.*
>
> *As of early 2011, the under-development version of CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors. This is not for lack of trying: we have devoted about six CPU-years to the task.*

Some history of Proof Assistants

- Church 1940: λ-calculus, simple type theory, higher order logic
- Curry Howard (De Bruijn): Formulas-as-Types
  Interpret formulas as types,
  Encode proofs as terms
  Proof-checking = Type-checking
- Automath (De Bruijn 1970s): first implementation of these ideas
- LCF (Milner), ML
- Coq, Hol, Isabelle, Lean, Agda, Mizar, PVS, ACL2, ...

These lectures

- Untyped lambda calculus next hour
  See the notes by Barendregt & Barendsen.
- Working with the Proof Assistant Coq
- Type Theory
  Is not only used for Proof Assistants but als very much in Programming Languages. In the lectures I'll devote attention to this. (Type checking algorithm, ...)