An Introduction to Proof Assistants

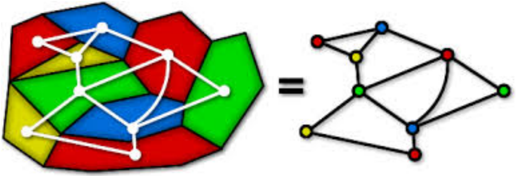
Student Seminar in Combinatorics: Mathematical Software

Patrick Schnider

ETH Zürich, 9. December 2014

Outline

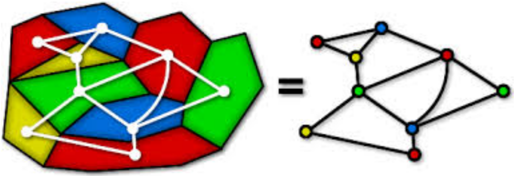
The Four-colour Theorem



Theorem: Every planar graph allows a proper vertex colouring with four colours.

1852: Posed by Francis Guthrie to his former Professor Augustus de Morgan.
1879: False Proof by Alfred Kempe.
1890: Percy Heawood finds mistake in Kempe's proof, shows Five-colour theorem.

The Four-colour Theorem



Theorem: Every planar graph allows a proper vertex colouring with four colours.

1976: Computer-assisted proof by Kenneth Appel and Wolfgang Haken. 1936 cases checked by a computer in over thousand hours.
1996: Easier Computer-assisted-proof by Neil Robertson, Daniel P. Sanders, Paul Seymour, and Robin Thomas. "Only" 633 cases.
2005: Proof formalized in the Proof Assistant Coq. No more need to trust the programs used to check the cases, only need to trust the Coq Kernel.

The Kepler Conjecture

Conjecture: No arrangement of equally sized spheres filling space has a greater average density than that of the cubic close packing and hexagonal close packing arrangements.

1611: Posed by Johannes Kepler.
1831: Proof for spheres arranged in a lattice by Carl Friedrich Gauss.
1900: Included in David Hilbert's "twenty three unsolved problems of mathematics".
1953: Laszlo Fejes-Toth: Proof by exhaustion in principle possible.

The Kepler Conjecture

Conjecture: No arrangement of equally sized spheres filling space has a greater average density than that of the cubic close packing and hexagonal close packing arrangements.

1998: Proof by Tom Hales. Involves solving around 100'000 linear programs. Annals of Mathematics: "99% certain of correctness, but cannot certify correctness of all computer calculations."
2003: Start of Flyspeck Program to formalize Proof with HOL and Isabelle.
2014: Flyspeck Program announced to be complete.

Formal Proofs

A formal proof is a finite sequence of sentences. Each sentence is either an axiom or follows from the preceding sentences. The last sentence in the sequence is a theorem.

Can be checked by computers effectively.

However, finding formal proofs is in general very hard.

Proof Assistants

A proof assistant is a software that **interacts** with the user to find formal proofs.

Not to be confused with automatic theorem provers.

Depending on the assistant, certain tasks are automated.

What Proof Assistants are there?

- HOL light
- Isabelle
- Coq
- Mizar (*)
- and many more...

(*) offers no automated tools, but extensive library.

What Proof Assistants are not

Not automated theorem provers, the user interaction is required! In fact the user has to do quite a lot.

Not tools to compute solutions for complicated problems! $\rightsquigarrow$ numerical software, computer algebra, many of the programs presented in other talks of this seminar...

Coq

Developed by INRIA in France, first release in 1989.

Written in OCaml.

The interaction with Coq is in Gallina.

Logical formalism is Calculus of inductive constructions (CIC).

Available for all major platforms.

Graphical interfaces: CoqIDE and ProofGeneral.

Lots of libraries and proof tactics.

Using Coq

Let's take a look at some examples!

Implementation Overview

The implementation of Coq is based on 8 parts:

Part	**Function**
1. The logical framework	Meta-language for terms of CIC
2. The language of constructions	language for CIC
3. The type-checker (Kernel)	checks formal proofs
4. The proof engine	interactive proof construction
5. The language of tactics	library of pre-implemented tactics
6. The vernacular interpreter	Interpreter of Gallina inputs
7. The parser and pretty-printer	Translation strings $\leftrightarrow$ formulas
8. The standard library	pre-implemented modules

Implementation Overview

The implementation of Coq is based on 8 parts:

Part	**Function**
1. The logical framework	Meta-language for terms of CIC
2. The language of constructions	language for CIC
3. The type-checker (Kernel)	checks formal proofs
4. The proof engine	interactive proof construction
5. The language of tactics	library of pre-implemented tactics
6. The vernacular interpreter	Interpreter of Gallina inputs
7. The parser and pretty-printer	Translation strings $\leftrightarrow$ formulas
8. The standard library	pre-implemented modules

Kernel: the de Bruijn Criterion

A proof assistant satisfies the de Bruijn criterion if it generates proofs that can be checked (independently of the system that created it) using a simple program (that a skeptical user can write him/herself).

In Coq, the Kernel is independent of the rest of the system and relatively small. There are only 5 rules in CIC to be checked.

Tactics

Tactics can also be programmed or extended by user.

Tactics can call other tactics.

Primitive tactics: Introducing variables, changing terms into equivalent terms,...
Defined tactics: Combination of primitive tactics.

Let us take a look at the source code of the tactic tauto.

Can we really trust Proof assistants?

In his paper "Flyspecking Flyspeck" Mark Adams mentions seven concerns:

1. Has a final theorem actually been proved in the assistant?
2. Does the final statement really mean what we think it means?
3. Were any axioms added that make the proof assistants theory inconsistent?
4. Are the settings for displaying concrete syntax configured in a way that happen to make a statement get misinterpreted?
5. Can we trust the proof assistant to correctly record and display all the information required for the review? (Pollack-inconsistency)
6. Is the proof assistant sound?
7. Is there a proof script that could make the proof assistant unsound?

Also, any auditor must assume malicious intent.

Pollack-inconsistency

Parser: parse: string $\to$ formula (Input)
Printer: print: formula $\to$ string (Output)
We would like to have $\mathrm{parse}(\mathrm{print}(\Phi)) = \Phi$.
In practice, this sometimes breaks.
Pollack-axioms: $\Phi_1 \Leftrightarrow \Phi_2$ when $\mathrm{print}(\Phi_1) = \mathrm{print}(\Phi_2)$.
A proof assistant is called Pollack-inconsistent if False is provable from Pollack-axioms.

HOL light, Isabelle, Coq and Mizar are all Pollack-inconsistent!

Conclusion

Questions?

References

- Wikipedia:
  - http://en.wikipedia.org/wiki/Four_color_theorem
  - http://en.wikipedia.org/wiki/Kepler_conjecture
  - http://en.wikipedia.org/wiki/Proof_assistant
  - http://en.wikipedia.org/wiki/Formal_proof
  - http://en.wikipedia.org/wiki/Coq
- Coq Homepage: https://coq.inria.fr/
- Coq References:
  - Coq Reference Manual
  - "Coq in a hurry": https://cel.archives-ouvertes.fr/inria-00001173v5/document
  - "Theorem Proving with Coq": http://flint.cs.yale.edu/cs430/sectionNotes/section1/CoqTutorial.pdf
  - Book "Software Foundations", B.Pierce et al: http://www.cis.upenn.edu/ bcpierce/sf/current/index.html

References (continued)

- Presentation by H. Geuvers: http://www.cs.ru.nl/ herman/ictopen.pdf
- Coq Source Code repository: https://gforge.inria.fr/git/coq/coq.git
- Mark Adams: Flyspecking Flyspeck. In: Mathematical Software - ICMS 2014.
- T.C. Hales: Computational Discrete Geometry. In: Mathematical Software - ICMS 2010.
- G. Gonthier: A computer-checked proof of the four colour theorem. http://research.microsoft.com/en-us/um/people/gonthier/4colproof.pdf