Conference *Computation and Complexity*
The Center for Algorithms and Interactive Scientific Software (CAISS)
The City College of New York

Computer-Aided Proofs and Their Significance

Sergei Artemov

(*The City University of New York - Graduate Center*)

May 12, 2006

The plan of this talk

- Proofs and Provability: basic observations
- General purpose proof assistants and their uses
- Metatheory of verification
- Incorporating decision procedures
- Four Color Theorem, Kepler's conjecture
- Formal proofs outside mathematics
- Conclusions

Proofs and Provability

Two closely connected functions of proofs:

- *truth certification*
- *providing insights* about the structure of objects

We will begin by discussing the former.

Given a sentence, find its proof:

- undecidable for general purpose quantified languages
- unfeasible for general propositional languages

Given S and p, certify that p is a proof of S:

- decidable and feasible for many general purpose languages
- practical, implemented in a variety of computer-based proof assistants

General purpose proof assistants

Prehistory: de Bruijn's Automath Project

Modern architecture: Robin Milner (1972) Stanford LCF (*Logic for Computable Functions*). Circa 1979 - Edinburgh's LCF - tactics, isolated trusted core, proof checker.

Current (incomplete) list: HOL, Coq, Mizar, Isabelle, PVS, Nuprl/MetaPRL .

Most use a goal-driven derivation: the user starts from the goal and "decomposes" (refines) it down to axioms and/or established facts (top-down derivation). At every moment, a partial derivation is a tree with possible ungrounded leaves. It becomes complete when all leaves are ground.

HOL

Stands for (classical) Higher-Order Logic, uses predicate calculus with terms from typed λ-calculus.

Mike Gordon (Cambrige University), 1988, a direct descendant of Edinburgh LCF. Current versions: (HOL88, HOL90, HOL98, HOL Light, HOL 4).

Mathematics formalized in HOL: real analysis up to fundamental theorem of calculus, complex numbers up to fundamental theorem of algebra, weak form of the Prime Number Theorem, floating-point arithmetic, etc.

HOL Light was designed by John Harrison and Konrad Slind, runs on standard PC's, and supports both top-down and down-top derivations. It has been used in the Flyspeck project to machine-check Tom Hales's proof of the Kepler conjecture. Success so far: the Jordan Curve Theorem.

Coq

Coq, INRIA, is based on Coquand's Calculus of Inductive Constructions (1985), extension of Girard's polymorphic λ-calculus F_ω. Its main goal was specification and verification of programs. Coq's basic logic is intuitionistic, and it includes a mechanism for automatic generation of certified programs from proofs of their specifications

Coq is widely used for formalization of mathematics: real analysis, constructive category theory, elements of constructive geometry, group theory, domain theory, fundamental group theory. A recent success story: formalization and verification of a proof of the Four Color Theorem (1999/2004).

Mizar

Non-interactive proof-checker, forward style from axioms to goals.
Started in 1974 (Andrzej Trybulec) as software to support a working mathematician in preparing papers.
Logic: classical first-order, Jaskowski's natural deduction.
Mathematics: Tarski-Grothendieck set theory.

Journal *Formalized Mathematics (a computer assisted approach)* established in 1990 and devoted solely to the formalizations of mathematics in Mizar. All papers are checked by the Mizar. They formalized the Jordan Curve Theorem for special polygons. They are in the process of formalizing general Jordan Curve Theorem. Mizar Mathematical Library includes 926 articles written by 175 authors and 41525 theorems, 7838 definitions, 722 schemes, 6805 registrations, 5784 symbols, 1903 keywords.

Isabelle

Isabelle (started in 1986, Larry Paulson, Cambridge University, and Tobias Nipkow, TU Munich), rather a logical framework ("generic proof assistant"), not tightly bound to one specific logic.

Meta-logic is intutionistic higher-order logic with equality; different logical systems can be defined: HOL, FOL, ZF, HOL with Scott's Logic for Computable Functions (domain theory) added, small fragment of Martin-Löf's Type Theory (ITT), Barendregt's Lambda Cube, and others.

Large theory library: elementary number theory (for example, Gauss's law of quadratic reciprocity), analysis (basic properties of limits, derivatives, and integrals), algebra (up to Sylow's theorem), and set theory (the relative consistency of the Axiom of Choice), the Prime Number Theorem.

PVS

Stands for *Prototype Verification System*, SRI International, commenced in 1990, intended for significant applications. PVS is a research prototype: it evolves and improves as the stress of real use exposes new requirements.

Based on simply typed classical higher-order logic extended with subtyping, dependent typing, and parametric theories which makes it somewhat closer to Coq and Nuprl.

Mathematical library: calculus, domain theory, program semantics, graph theory, a very elaborate library of decision procedures used for hardware and software verification.

NuPRL

PRL = Proof Refinement Logic, 1973, Nu = ν, a version indicator.
NuPRL appeared around 1984, Robert Constable, Cornell, now versions 1-5.
Built around Martin-Löf's Type Theory (ITT), a higher-order intuitionistic system. Aimed at program specification and verification, has an impressive list of successes. Nuprl is also a direct descendant of Edinburgh LCF.

Formalized mathematical theories including but not limited to constructive real analysis, computational abstract algebra (multivariate polynomial arithmetic, unique factorization domains), extracting constructive content from classical proofs (Higman's Lemma), automata theory, Turing machines, etc. Some major protocol verification successes.

MetaPRL

Most recent development, 1998, Jason Hickey, joined by Aleksey Nogin, Cornell, Caltech, CUNY, Moscow, HRL Laboratories, a direct descendant of NuPRL, address scalability and modularity limitations of the latter. MetaPRL is tested to be over 100 times faster than Nuprl in major domains.

MetaPRL is a logical framework with its main effort invested in Constructive Type Theory (CTT); it also supports first-order logic, Aczel's constructive set theory CZF, and others.

Currently is used in designing verifiable compilers. Some noticeable mathematical formalizations: abstract algebra for CTT and CZF.

Metatheory of mathematics. Completeness is an issue:

Assume T is consistent.

Then T does not prove its own consistency.

Metatheory of verification. Soundness is the main concern:

Assume T is consistent.

Why bother whether or not T proves its own consistency?

Trusted Core (TC) = basic proof checker with a small, transparent code. Tested extensively, much more than the mind of an individual mathematician could ever be. Tradeoff - it is infeasible to write all the proofs in a script acceptable by TC (elementary proofs). Expansions of TC are needed: tactics (rules), decision procedures.

The main metamathematical problem here:

Does a proof in an expanded system yield a proof script checkable by TC?

Metatheory of verification: (Davis & Schwartz, 1979)

Verified in V tactics: Rule Γ/φ such that V proves

$$\mathit{Provable}(\Gamma) \to \mathit{Provable}(\varphi)$$

Stability: Any use of verified tactics yields an elementary proof

For any verified rule Γ/φ, if $V + \Gamma/\varphi \vdash \psi$, then $V \vdash \psi$.

For establishing stability of V, one needs to trust more then V itself.

Indeed, when V tries to emulate the rule Γ/φ, then the following happens:

$$V \vdash \Gamma \;\Rightarrow\; V \vdash \mathit{Provable}(\Gamma) \;\Rightarrow\; V \vdash \mathit{Provable}(\varphi) \;\Rightarrow (?)\; V \vdash \varphi.$$

To conclude that $V \vdash \varphi$, a reflection principle in V is needed:

$$\mathit{Provable}(\mathit{Provable}(\varphi)) \to \mathit{Provable}(\varphi)\ ,$$

which is not provable in V.

Metatheory of verification: a more delicate analysis (Artemov, 1999).

Explicit proofs: $t{:}\varphi \sim$ *t is a proof of* φ.

Explicit reflection $t{:}\varphi \to \varphi$ is internally provable.

Explicitly verified in V rule Γ/φ: there is a total computable function f such that V proves

$$x{:}\Gamma \quad \Rightarrow \quad f(x){:}\varphi$$

Explicit stability: $V + \Gamma/\varphi = V$ for each explicitly verified Γ/φ.

Explicit stability of V can be justified in V: just apply the proof checker.

$$V \vdash \Gamma \;\Rightarrow\; V \vdash t{:}\Gamma \;\Rightarrow\; V \vdash f(t){:}\varphi \;\Rightarrow\; V \vdash \varphi \;.$$

For constructive systems, stability yields explicit stability (explicit definability, independence of premises). Other good cases: derivable rules, provable rules ($V \vdash \Gamma \to \varphi$).

Moral so far

Proof assistants are considered safe, if they produce an elementary proof checked by the trusted core. Elaborate system of tactics (lemmas, rules) provide a comfortable level of flexibility and extendability.

Incorporating decision procedures

Useful decision procedures that decide whether $P(x)$ holds, but do not necessarily produce proofs of $P(x)$ directly: deciding tautologies, validity of Presburger liner terms, reducibility of configurations, etc: fast and potentially unreliable. How to convert them into tactics?

Constable et al, Boulton, Harrison, Norrish, Kreitz, Schmitt, etc.

Some important techniques:

- Proof by evidence
- Proforma theorem
- Proof script
- etc.

Proof by evidence

A ‘dirty’ decision procedure produces a result which is easy to verify directly:

- a solution of a system of equations
- a circular chain of inequalities
- a satisfying substitution
- etc.

Using such a procedure is safe, given the system checks the result and builds its formal proof.

Proforma theorem

The system formally proves the correctness of a given decision procedure:
for each possible input x, if the decision procedure accepts $P(x)$, then there is a formal proof of $P(x)$.
Proforma theorems then can be used as lemmas in the final formal proof.

Example. A completeness theorem:
if a given satisfiability test fails for 'not φ,' then there is a proof of φ.

Actually, an explicit version of this theorem is needed:
We produce a function and prove that it transforms a given failed run of a satisfiability test for φ into a proof of φ.

Proforma theorems are difficult to establish. Reflection is involved. We have to execute the reflected version of the decision procedure from the Proforma theorem, which is often less efficient, than its original version.

Proof script

Sometimes it is too difficult to prove proforma theorem but relatively easy to prove its instances for each particular problem.

For example, proforma theorem might use a very complicated inductive reasoning, but for each particular problem, one can avoid induction using iteration of similar reasoning steps.

One builds a function which transforms a result of a decision procedure for φ into a proof of φ, and leaves it to the general proof checker to verify the resulting proofs on a case-by-case basis.

This approach seems easier then the previous one, but it could be less efficient, since we have to actually produce an elementary proof for each case.

Four Color Theorem

Since 1852 - Guthrie, de Morgan, Peirce, Hamilton, Cayley, Birkhoff, Lebesgue.

Appel and Haken, 1976, provided a proof that involved

- initial manual case analysis of about 10,000 cases,
- a computer analysis of a billion cases.

Scepticism: programming is known to be error-prone, difficult to relate precisely to specifications, small errors in the manual analysis.

Robertson, Sanders, Seymour, Thomas, 1995, a more streamlined version using a similar argument, but employing a computer to check both the large and the giant case analyses, made the former four times smaller. The question of verifying the computer programs remained.

Four Color Theorem continued

December 2004, Georges Gonthier (Microsoft Research Cambridge) announced a successful formalization of the 4CT, which is fully checked by the Coq v7.3.1 proof assistant. It is largely based on Robertson *et al*, but the two weakest links of the proof have been removed:

- the manual verification of combinatorial arguments,
- the manual verification that the custom computer programs correctly fill in parts of those arguments.

A formal proof script (60,000 lines) covers both the mathematical and computational parts of the proof. This script was then run through the Coq proof checking system (3 days). The programs used have been supplied by the corresponding Proforma-style theorems. A checkable proof witness can be produced: a huge higher-order λ-term containing detailed description of all logical steps.

Kepler's conjecture

In 1611, Kepler proposed that the natural close packing (actually, either of the two) is the densest possible sphere packing, and this assertion is known as *Kepler's Conjecture, KC*.

In 1831, Gauss proved KC for regular packings.

In 1900, Hilbert included KC in his famous list of problems (Problem 18).

In 1953, Tóth showed that the problem of determining the maximum density of all arrangements (regular and irregular) could be reduced to a finite (but very large) number of calculations.

In 1998, Thomas Hales, then at the University of Michigan, announced a proof consisting of 250 pages of notes and 3 gigabytes of computer programs, data, and results.

Hales's proof

Following Tóth, Thomas Hales determined that KC can be solved by minimizing a function with 150 variables over 5,000 different configurations of spheres which involved solving approximately 100,000 linear programming problems.

Annals of Mathematics appointed a panel of twelve referees. In 2003, after four years of work, the panel reported that they were "99% certain" of the correctness of the proof, but they could not certify the correctness of the computer calculations.

In 2003, Hales announced a project *Flyspeck* to produce a complete formal proof of KC: "I have no other choice." Hales picked HOL Light, initial estimates of time: four years. Current estimates: 20 years total to formalize the proof, which can, however, be reduced by parallel work.

Some results of Flyspeck

November 2005, *Annals of Mathematics* published the paper

Hales, T. C. "A Proof of the Kepler Conjecture."
Annals of Mathematics, v. 162, pp. 1065-1185, 2005.
http://www.math.princeton.edu/∼annals/issues/2005/Nov2005/Hales.pdf.

The computational portions will be published in *Discrete and Computational Geometry*.

Some famous computer bugs

London ambulance system (1992). A succession of software engineering failures, especially in project management, caused two failures of London's (England) ambulance dispatch system. The repair cost was estimated at £9m, but it is believed that people died who would not have died if ambulances had reached them as promptly as they would have without the failures.

Pentium FDIV bug (1994). Cost Intel half a billion, and a lot of agony on the way to an eventual no-strings-attached recall.

Ariane 5 (1996). The Ariane 5 rocket exploded on its maiden flight in June 4, 1996 because the navigation package was inherited from the Ariane 4 without proper testing.

USS Yorktown (1998). A crew member of the guided-missile cruiser USS Yorktown mistakenly entered a zero for a data value, which resulted in a division by zero. The error cascaded and eventually shut down the ship's propulsion system. The ship was dead in the water for several hours because a program didn't check for valid input.

Mars Climate Orbiter (1999). The 125 million dollar Mars Climate Orbiter was lost by NASA. One of the development teams used Imperial measurement while the other used the metric system of measurement.

Bugs cost about $60 billion annually in the US alone. About a third of that cost could be eliminated by improving testing.

Formal proofs outside mathematics? In fact, a vast majority of them.

All proof assistants mentioned (but, perhaps, Mizar) have been targeting verification applications, all have impressive success records.

Massive hires of formal method experts by industry. Harrison (HOL light) is now Intel's senior engineer.

In programming languages the state of the art is almost at the point where an electronic appendix with machine-checked proofs accompanying papers is fast becoming the norm.

Dual approach to proving, model checking. These two approaches cooperate and complement each other.

Conclusions

Computer-aided proofs are playing an increasingly prominent role.

Computers bring precision to proof building. Computer-verified proofs are more reliable than those verified by a human mathematician.

Proof assistants are sometimes the only tool capable of handling an increasing mathematical complexity beyond the capacity of any human being.

New layer of mathematical and technological challenges in this area.

Reflection mechanisms are vital for heavy duty tasks in formal verification.

It takes a different set of skills to formalize a long proof than to find one.