Automated Reasoning and its Applications

John Harrison

Intel Corporation

Colloquium, Institute of Mathematics

Hanoi

30th July 2009

What is automated reasoning?

Attempting to perform logical reasoning in an automatic and algorithmic way. An old dream:

- Hobbes (1651): “*Reason* . . . is nothing but *reckoning* (that is, adding and subtracting) of the consequences of general names agreed upon, for the *marking* and *signifying* of our thoughts.”
- Leibniz (1685) “When there are disputes among persons, we can simply say: Let us calculate [calculemus], without further ado, to see who is right.”

Nowadays, by ‘automatic and algorithmic’ we mean ‘using a computer program’.

What does automated reasoning involve?

There are two steps to performing automated reasoning, as anticipated by Leibniz:

- Express *statement* of theorems in a formal language. (Leibniz's *characteristica universalis*.)
- Use automated algorithmic manipulations on those formal expressions. (Leibniz's *calculus ratiocinator*).

Is that really possible?

Theoretical and practical limitations

- Modern results in logic (Gödel, Tarski) imply that not even elementary number theory can be done completely automatically.
- There *are* formal proof systems (e.g. first-order set theory) and semi-decision procedures that will in principle find the proof of anything provable in ‘ordinary’ mathematics.
- In practice, because of time or space limits, these automated procedures are not all that useful, and we may prefer an interactive arrangement where a human guides the machine.

Why automated reasoning?

For general intellectual interest? It is a fascinating field that helps to understand the real nature of mathematical creativity. Or more practically:

- To check the correctness of proofs in mathematics, supplementing or even replacing the existing ‘social process’ of peer review etc. with a more objective criterion.
- To extend rigorous proof from pure mathematics to the verification of computer systems (programs, hardware systems, protocols etc.), supplementing or replacing the usual testing process.

These are currently the two main drivers of progress in the field.

Automated Reasoning is not the same as Computer Algebra

Both systems for symbolic computation, but rather different:

- Theorem provers are more logically flexible and rigorous
- CASs are generally easier to use and more efficient/powerful

Some systems like MathXpert, Theorema blur the distinction somewhat . . .

Expressivity of logic

English	Formal
false	$\bot$
true	$\top$
not p	$\neg p$
p and q	$p \wedge q$
p or q	$p \vee q$
p implies q	$p \Rightarrow q$
p iff q	$p \Leftrightarrow q$
for all x, p	$\forall x.\ p$
there exists x such that p	$\exists x.\ p$

Limited expressivity in CASs

Often limited to conditional equations like

$$\sqrt{x^2} = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x \leq 0 \end{cases}$$

whereas using logic can say many interesting (and highly undecidable) things

$$\forall x \in \mathbb{R}.\ \forall \epsilon > 0.\ \exists \delta > 0.\ \forall x'.\ |x - x'| < \delta \Rightarrow |f(x) - f(x')| < \epsilon$$

Unclear expressions in CASs

Consider an equation $(x^2 - 1)/(x - 1) = x + 1$ from a CAS. What does it mean?

- Universally valid identity (albeit not quite valid)?
- Identity true when both sides are defined
- Identity over the field of rational functions
- . . .

Lack of rigour in many CASs

CASs often apply simplifications even when they are not strictly valid.

Hence they can return wrong results.

Consider the evaluation of this integral in Maple:

$$\int_0^\infty \frac{e^{-(x-1)^2}}{\sqrt{x}} dx$$

We try it two different ways:

An integral in Maple

```
> int(exp(-(x-t)ˆ2)/sqrt(x), x=0..infinity);
```

$$\frac{1}{2}\,\frac{e^{-t^2}\left(-\frac{3(t^2)^{\frac{1}{4}}\,\pi^{\frac{1}{2}}\,2^{\frac{1}{2}}\,e^{\frac{t^2}{2}}\,K_{\frac{3}{4}}(\frac{t^2}{2})}{t^2}+(t^2)^{\frac{1}{4}}\,\pi^{\frac{1}{2}}\,2^{\frac{1}{2}}\,e^{\frac{t^2}{2}}\,K_{\frac{7}{4}}(\frac{t^2}{2})\right)}{\pi^{\frac{1}{2}}}$$

```
> subs(t=1,%);
```

$$\frac{1}{2}\,\frac{e^{-1}\left(-3\pi^{\frac{1}{2}}\,2^{\frac{1}{2}}\,e^{\frac{1}{2}}\,K_{\frac{3}{4}}(\frac{1}{2})+\pi^{\frac{1}{2}}\,2^{\frac{1}{2}}\,e^{\frac{1}{2}}\,K_{\frac{7}{4}}(\frac{1}{2})\right)}{\pi^{\frac{1}{2}}}$$

```
> evalf(%);
```

$$0.4118623312$$

```
> evalf(int(exp(-(x-1)ˆ2)/sqrt(x), x=0..infinity));
```

$$1.973732150$$

Orientation

Can divide theorem proving research into the following streams:

- Fully automated theorem proving
  - AI-oriented
  - Logic-oriented
- Interactive theorem proving
  - Verification-oriented
  - Mathematics-oriented

Early research in automated reasoning

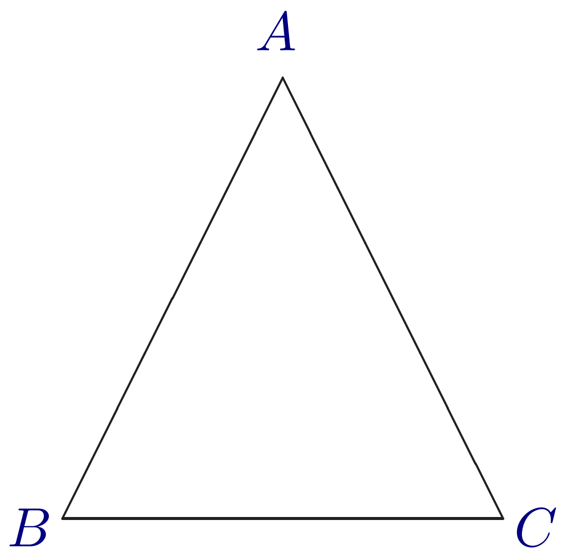
Most early theorem provers were fully automatic, even though there were several different approaches:

- Human-oriented AI style approaches (Newell-Simon, Gelerntner)
- Machine-oriented algorithmic approaches (Davis, Gilmore, Wang, Prawitz)

Modern work dominated by machine-oriented approach but some successes for AI approach.

A theorem in geometry (1)

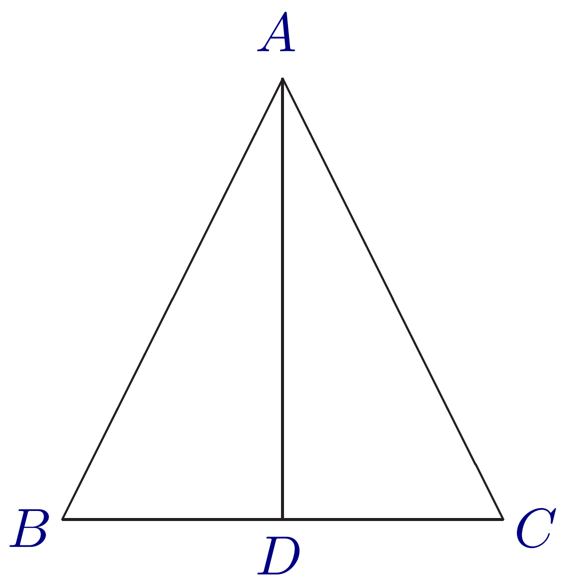
Example of AI approach in action:



If the sides AB and AC are equal (i.e. the triangle is isosceles), then the angles ABC and ACB are equal.

A theorem in geometry (2)

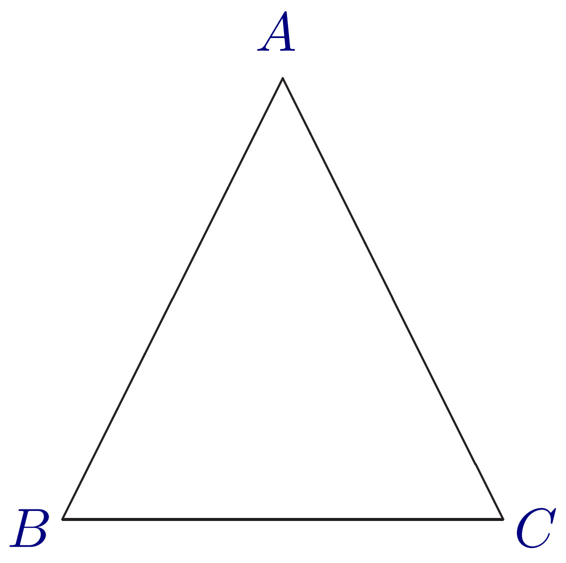
Drop perpendicular meeting BC at a point D:



and then use the fact that the triangles ABD and ACD are congruent.

A theorem in geometry (3)

Originally found by Pappus but not in many books:



Simply, the triangles ABC and ACB are congruent.

The Robbins Conjecture (1)

Huntington (1933) presented the following axioms for a Boolean algebra:

$$\begin{aligned} x + y &= y + x \\ (x + y) + z &= x + (y + z) \\ n(n(x) + y) + n(n(x) + n(y)) &= x \end{aligned}$$

Herbert Robbins conjectured that the Huntington equation can be replaced by a simpler one:

$$n(n(x + y) + n(x + n(y))) = x$$

The Robbins Conjecture (2)

This conjecture went unproved for more than 50 years, despite being studied by many mathematicians, even including Tarski.

It because a popular target for researchers in automated reasoning.

In October 1996, a (key lemma leading to) a proof was found by McCune's program EQP.

The successful search took about 8 days on an RS/6000 processor and used about 30 megabytes of memory.

Interactive theorem proving

The idea of a more ‘interactive’ approach was already anticipated by pioneers, e.g. Wang (1960):

> [...] the writer believes that perhaps machines may more quickly become of practical use in mathematical research, not by proving new theorems, but by formalizing and checking outlines of proofs, say, from textbooks to detailed formalizations more rigorous that *Principia* [Mathematica], from technical papers to textbooks, or from abstracts to technical papers.

However, constructing an effective combination is not so easy.

The 17 Provers of the World

Freek Wiedijk's book *The Seventeen Provers of the World* (Springer-Verlag lecture notes in computer science volume 3600) describes:

HOL, Mizar, PVS, Coq, Otter/IVY, Isabelle/Isar, Alfa/Agda, ACL2, PhoX, IMPS, Metamath, Theorema, Lego, Nuprl, Omega, B prover, Minlog.

Each one has a proof that $\sqrt{2}$ is irrational.

There are many other systems besides these . . .

Effective interactive theorem proving

What makes a good interactive theorem prover?

- Reliability
- Library of existing results
- Intuitive input format
- Powerful automated steps
- Programmability
- Checkability of proofs

The various systems have different strengths and weaknesses when considered according to these criteria.

Benefits and costs

Working in an interactive theorem prover offers two main benefits:

- Confidence in correctness (if theorem prover is sound).
- Automatic assistance with tedious/routine parts of proof.

However, formalization and theorem proving is hard work, even for a specialist. Mainly used in:

- Formal verification of computer systems
- Formalization of pure mathematics

The human cost of computer bugs

Computers are often used in safety-critical systems where a failure could cause loss of life.

- Heart pacemakers
- Aircraft
- Nuclear reactor controllers
- Car engine management systems
- Radiation therapy machines
- Telephone exchanges (!)
- ...

Financial cost of bugs

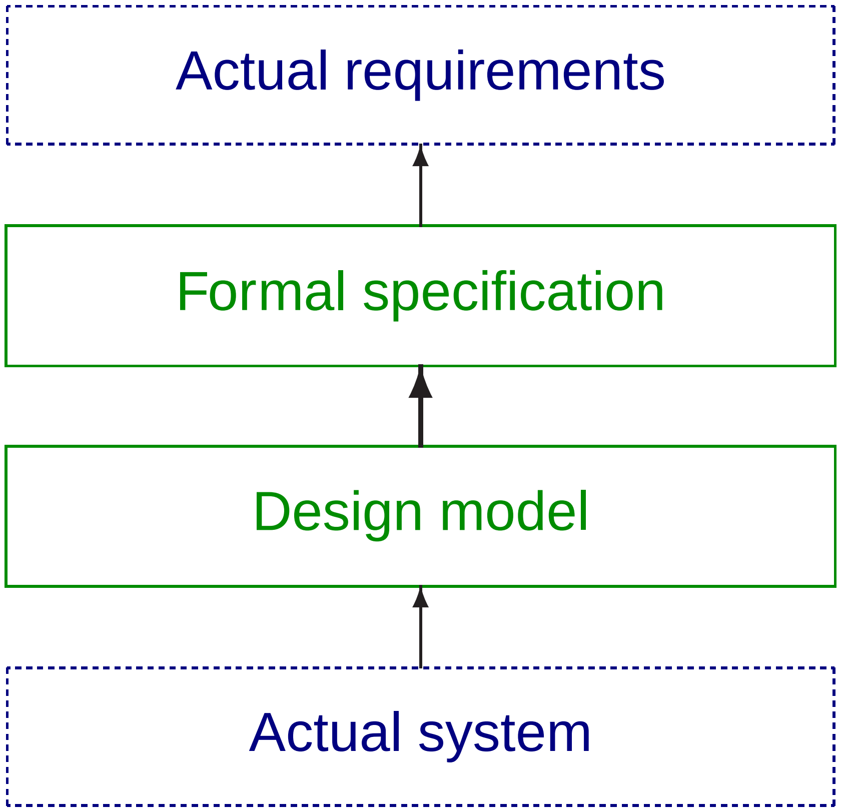
Even when not a matter of life and death, bugs can be financially serious if a faulty product has to be recalled or replaced.

- 1994: floating-point division (FDIV) bug in the Intel®Pentium® processor (cost $\$500M$).
- 1996: floating-point conversion overflow causes self-destruct of Ariane 5 rocket on its maiden flight (rocket and payload another $\$500M$).

Formal verification using computer theorem provers is now routinely used for parts of such critical systems, and we expect to see more of this in the future.

Formal verification

Formal verification: mathematically prove the correctness of a *design* with respect to a mathematical *formal specification*.



Verification vs. testing

Verification has some advantages over testing:

- Exhaustive.
- Improves our intellectual grasp of the system.

However:

- Difficult and time-consuming.
- Only as reliable as the formal models used.

Formal verification methods

Many different methods are used in formal verification, mostly trading efficiency and automation against generality.

- Propositional tautology checking
- Symbolic simulation
- Symbolic trajectory evaluation
- Temporal logic model checking
- Decidable subsets of first order logic
- First order automated theorem proving
- Interactive theorem proving

Logic and circuits

The correspondence between digital logic circuits and propositional logic has been known for a long time.

Digital design	Propositional Logic
circuit	formula
logic gate	propositional connective
input wire	atom
internal wire	subexpression
voltage level	truth value

Many problems in circuit design and verification can be reduced to automated propositional tautology or satisfiability checking (‘SAT’).

For example optimization correctess: $\phi \Leftrightarrow \phi'$ is a tautology.

Applying theorem provers in mathematics

Interactive theorem provers have been used to formalize many non-trivial theorems of mathematics

See `http://www.cs.ru.nl/~freek/100/` for examples, e.g.

- Jordan Curve Theorem — Tom Hales (HOL Light), Artur Kornilowicz et al. (Mizar)
- Prime Number Theorem — Jeremy Avigad et al (Isabelle/HOL), John Harrison (HOL Light)
- Four-colour theorem — Georges Gonthier (Coq)

These indicate that highly non-trivial results are within reach. However these all required months/years of work.

Are ordinary proofs in doubt?

Mathematical proofs are subjected to peer review, but errors often escape unnoticed.

> Professor Offord and I recently committed ourselves to an odd mistake (Annals of Mathematics (2) 49, 923, 1.5). In formulating a proof a plus sign got omitted, becoming in effect a multiplication sign. The resulting false formula got accepted as a basis for the ensuing fallacious argument. (In defence, the final result was known to be true.)

A book by Lecat gave 130 pages of errors made by major mathematicians up to 1900.

A similar book today would no doubt fill many volumes.

Most doubtful informal proofs

What are the proofs where we do in practice worry about correctness?

- Those that are just very long and involved. Classification of finite simple groups, Seymour-Robertson graph minor theorem
- Those that involve extensive computer checking that cannot in practice be verified by hand. Four-colour theorem, Hales's proof of the Kepler conjecture
- Those that are about very technical areas where complete rigour is painful. Some branches of proof theory, or formal properties of type systems

The discouraging history of the 4-colour Theorem

Early history indicates fallibility of the traditional social process:

- Proof claimed by Kempe in 1879
- Flaw only point out in print by Heaywood in 1890

Later proof by Appel and Haken was apparently correct, but gave rise to a new worry:

- How to assess the correctness of a proof where many explicit configurations are checked by a computer program?

Most worries finally dispelled by Gonthier's formal proof in Coq.

Hales’s Proof of the Kepler Conjecture

No packing of spheres in 3-dimensional space has higher density than the natural packing commonly used to stack oranges, cannonballs etc.

Tom Hales, working with Samuel Ferguson, proved this in 1998, with 300 pages of mathematics using the calculations performed by about 40,000 lines of computer code.

A panel of 12 referees for *Annals of Mathematics,* studied the proof for 4 years. They finally returned with the disappointing verdict that they were ‘99% certain’ of the correctness.

The proof was published in the *Annals*, but this process is somewhat unsatisfactory, and the same thing is likely to recur more and more often in the future.

The Flyspeck project

We should be able to do better by complete *formalization*, removing almost all practical possibility of errors.

Hales launched the Flyspeck Project to completely formalize the proof of the Kepler conjecture.

Considerable progress has already been made, and a workshop devoted to the project is just coming to an end here in Hanoi.

`http://weyl.math.pitt.edu/hanoi2009/`

If you want to get involved, see, e.g.

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

Conclusions

Let me finish with a quote from Tom Hales about the Flyspeck project

> In truth, my motivations for the project are far more complex than a simple hope of removing residual doubt from the minds of few referees. Indeed, I see formal methods as fundamental to the long-term growth of mathematics.