A Short Survey of Automated Reasoning

John Harrison

Intel Corporation

Algebraic Biology 2007

Linz

2nd July 2007

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

Theorem provers and computer algebra systems

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)^2)/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)^2)/sqrt(x), x=0..infinity));
```

$$1.973732150$$

Early research in automated reasoning

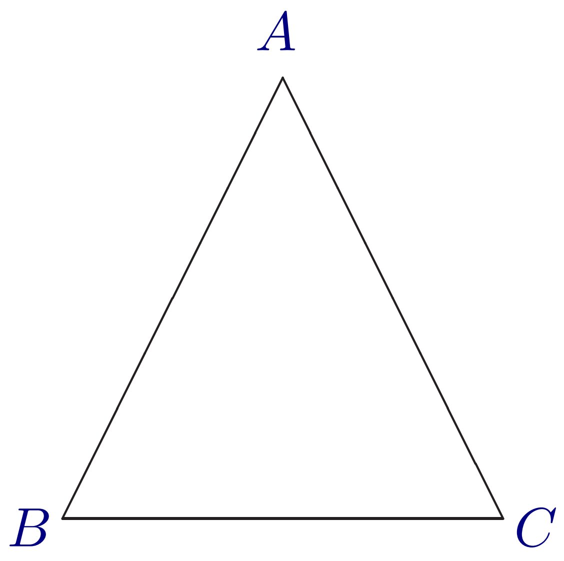
Most early theorem provers were fully automatic, even though there were several different approaches:

- 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)

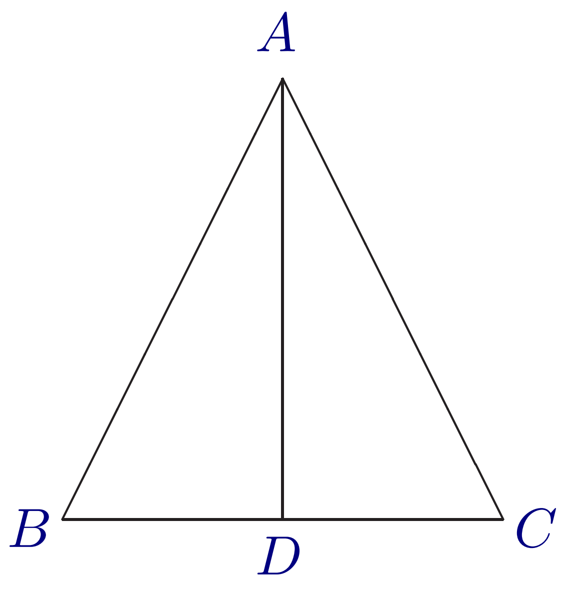
Example of AI approach in action:



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

A theorem in geometry (2)

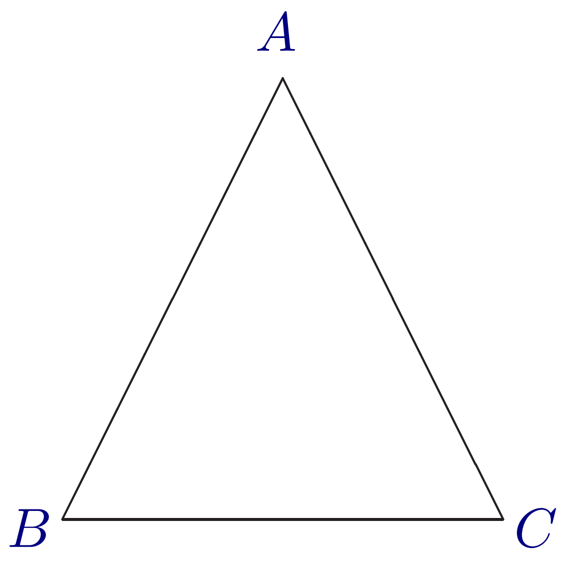
Drop perpendicular meeting BC at a point D:



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.

What can be automated?

- Validity/satisfiability in propositional logic is decidable (SAT).
- Validity/satisfiability in many temporal logics is decidable.
- Validity in first-order logic is *semidecidable*, i.e. there are complete proof procedures that may run forever on invalid formulas
- Validity in higher-order logic is not even *semidecidable* (or anywhere in the arithmetical hierarchy).

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 propositional tautology or satisfiability checking (‘SAT’).

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

Combinatorial problems

Many other apparently difficult combinatorial problems can be encoded as Boolean satisfiability (SAT), e.g. scheduling, planning, even factorization.

$$\begin{aligned}
\neg(\quad & (out_0 \Leftrightarrow x_0 \land y_0)\land \\
& (out_1 \Leftrightarrow (x_0 \land y_1 \Leftrightarrow \neg(x_1 \land y_0)))\land \\
& (v_2^2 \Leftrightarrow (x_0 \land y_1) \land x_1 \land y_0)\land \\
& (u_2^0 \Leftrightarrow ((x_1 \land y_1) \Leftrightarrow \neg v_2^2))\land \\
& (u_2^1 \Leftrightarrow (x_1 \land y_1) \land v_2^2)\land \\
& (out_2 \Leftrightarrow u_2^0) \land (out_3 \Leftrightarrow u_2^1)\land \\
& \neg out_0 \land out_1 \land out_2 \land \neg out_3)
\end{aligned}$$

Read off the factorization $6 = 2 \times 3$ from a refuting assignment.

Pure first-order logic

Though incomplete, automated theorem provers are often better at FOL than people, e.g.

$$
\begin{array}{l}
(\forall x\ y\ z.\ P(x,y) \land P(y,z) \Rightarrow P(x,z)) \land \\
(\forall x\ y\ z.\ Q(x,y) \land Q(y,z) \Rightarrow Q(x,z)) \land \\
(\forall x\ y.\ Q(x,y) \Rightarrow Q(y,x)) \land \\
(\forall x\ y.\ P(x,y) \lor Q(x,y)) \\
\Rightarrow (\forall x\ y.\ P(x,y)) \lor (\forall x\ y.\ Q(x,y))
\end{array}
$$

The need for theories

People usually use extensive background in set theory, arithmetic, algebra or geometry when they deem something ‘obvious’.

For example, the Mutilated Checkerboard . . .

In practice, we need to reason about theories or higher-order objects, which in general takes us well into the undecidable

Some arithmetical theories

- Linear theory of $\mathbb{N}$ or $\mathbb{Z}$ is decidable. Nonlinear theory not even semidecidable.
- Linear and nonlinear theory of $\mathbb{R}$ is decidable, though complexity is very bad in the nonlinear case.
- Linear and nonlinear theory of $\mathbb{C}$ is decidable. Commonly used in geometry.

Many of these naturally generalize known algorithms like linear/integer programming and Sturm's theorem.

Quantifier elimination

Many decision methods based on quantifier elimination, e.g.

- $\mathbb{C} \models (\exists x.\ x^2 + 1 = 0) \Leftrightarrow \top$
- $\mathbb{R} \models (\exists x. ax^2 + bx + c = 0) \Leftrightarrow a \neq 0 \wedge b^2 \geq 4ac \vee a = 0 \wedge (b \neq 0 \vee c = 0)$
- $\mathbb{Q} \models (\forall x.\ x < a \Rightarrow x < b) \Leftrightarrow a \leq b$
- $\mathbb{Z} \models (\exists k\ x\ y.\ ax = (5k + 2)y + 1) \Leftrightarrow \neg(a = 0)$

If we can decide variable-free formulas, quantifier elimination implies completeness.

Again generalizes known results like closure of constructible sets under projection.

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? Most agree on:

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

Several other characteristics are more controversial:

- Programmability
- Checkability of proofs

LCF

One successful solution was pioneered in Edinburgh LCF (‘Logic of Computable Functions’).

The same ‘LCF approach’ has been used for many other theorem provers.

- Implement in a strongly-typed functional programming language (usually a variant of ML)
- Make `thm` (‘theorem’) an abstract data type with only simple primitive inference rules
- Make the implementation language available for arbitrary extensions.

Gives a good combination of extensibility and reliability.

Now used in Coq, HOL, Isabelle and several other systems.

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.

Current niches

We currently see use of theorem proving where:

- The cost of error is too high, e.g. $475M for the floating-point bug in the Intel®Pentium® processor.
- There are genuine doubts in the community about the correctness of a proof, e.g. Hales's proof of the Kepler Conjecture.

Signs that theorem proving is starting to expand beyond these niches.

Conclusions

Automated reasoning is a well-established subfield of symbolic computation.

Some techniques are already making it to the mainstream (e.g. use of SAT for combinatorial problems).

Current interest comes mainly from those interested in formal verification and/or the formalization of mathematics.

Perhaps there are some new applications, e.g. in biology, waiting to be discovered?