# Formal Theorem Proving and Sum-of-Squares Techniques

John Harrison

Intel Corporation

LIDS Seminar, MIT

Fri 16th April 2010 (15:00-16:00)

## Orientation

Can divide theorem proving research into the following streams:

- Fully automated theorem proving
  - Human-oriented AI style approaches (Newell-Simon, Gelerntner)
  - Machine-oriented algorithmic approaches (Davis, Gilmore, Wang, Prawitz)
- 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 expressive/flexible and rigorous
- CASs are generally easier to use and more efficient/powerful

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

## Logical notation is very expressive

| 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 $p$ | $\exists x.\, p$ |

## 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).

## Applications

SAT has many applications such as

- Digital logic verification using the correspondence between circuits and formulas.
- Combinatorial problems such as scheduling.

Automated reasoning in first-order logic (even just equational logic) has seen some successes too, e.g. solution by McCune of the Robbins conjecture and other open problems.

## The need for theories

But 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.

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

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.
- A mathematical proof presents difficulties for the traditional peer review process, e.g. Hales's proof of the Kepler Conjecture.

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

## HOL Light overview

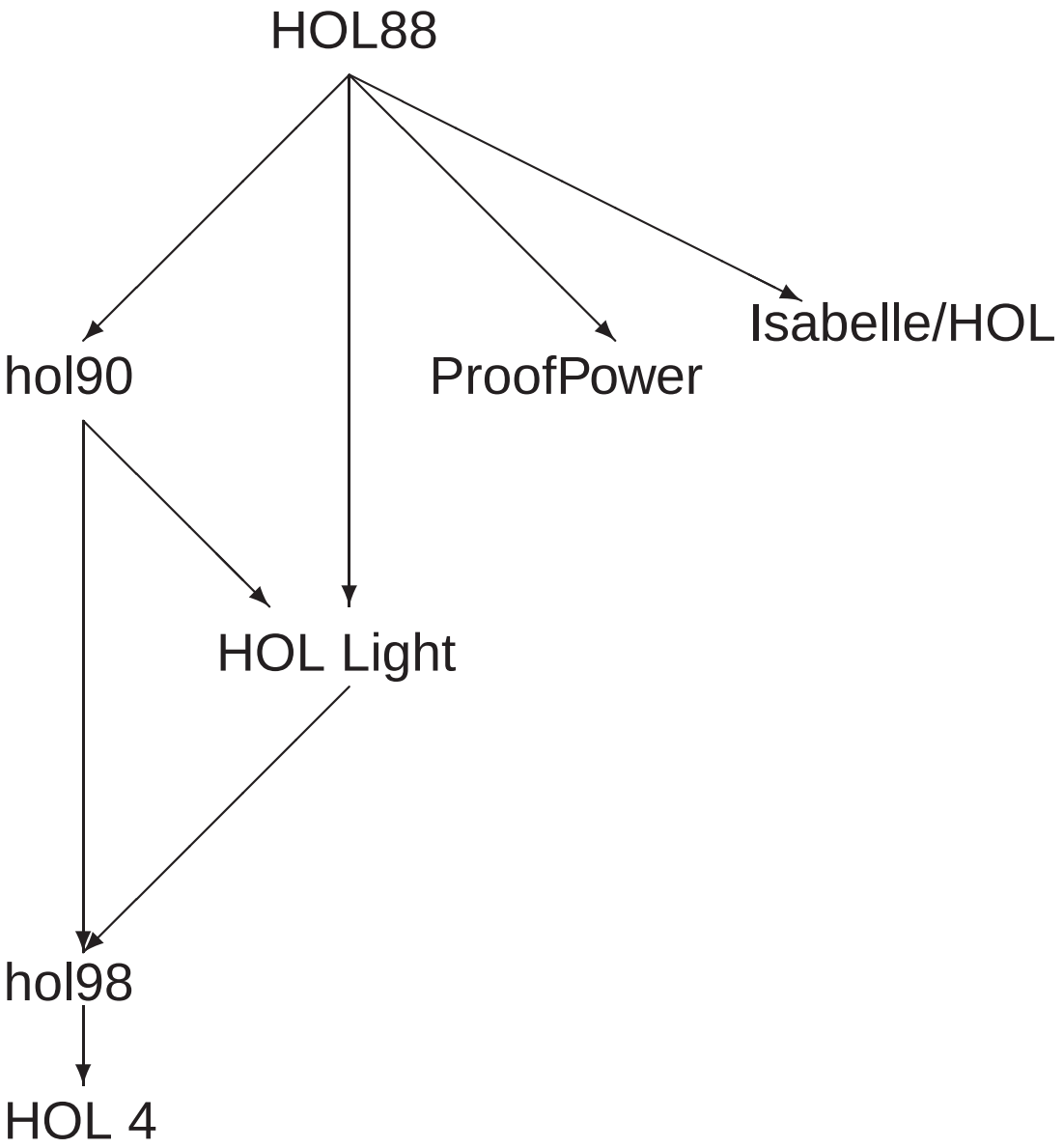
HOL Light is a member of the HOL family of provers, descended from Mike Gordon's original HOL system developed in the 80s.

An LCF-style proof checker for classical higher-order logic built on top of (polymorphic) simply-typed $\lambda$-calculus.

HOL Light is designed to have a simple and clean logical foundation.

Written in Objective CAML (OCaml).

## The HOL family DAG



## HOL Light primitive rules (1)

$$\frac{}{\vdash t = t}\ \texttt{REFL}$$

$$\frac{\Gamma \vdash s = t \quad \Delta \vdash t = u}{\Gamma \cup \Delta \vdash s = u}\ \texttt{TRANS}$$

$$\frac{\Gamma \vdash s = t \quad \Delta \vdash u = v}{\Gamma \cup \Delta \vdash s(u) = t(v)}\ \texttt{MK\_COMB}$$

$$\frac{\Gamma \vdash s = t}{\Gamma \vdash (\lambda x.\, s) = (\lambda x.\, t)}\ \texttt{ABS}$$

$$\frac{}{\vdash (\lambda x.\, t)x = t}\ \texttt{BETA}$$

## HOL Light primitive rules (2)

$$\frac{}{\{p\} \vdash p}\ \texttt{ASSUME}$$

$$\frac{\Gamma \vdash p = q \quad \Delta \vdash p}{\Gamma \cup \Delta \vdash q}\ \texttt{EQ\_MP}$$

$$\frac{\Gamma \vdash p \quad \Delta \vdash q}{(\Gamma - \{q\}) \cup (\Delta - \{p\}) \vdash p = q}\ \texttt{DEDUCT\_ANTISYM\_RULE}$$

$$\frac{\Gamma[x_1, \ldots, x_n] \vdash p[x_1, \ldots, x_n]}{\Gamma[t_1, \ldots, t_n] \vdash p[t_1, \ldots, t_n]}\ \texttt{INST}$$

$$\frac{\Gamma[\alpha_1, \ldots, \alpha_n] \vdash p[\alpha_1, \ldots, \alpha_n]}{\Gamma[\gamma_1, \ldots, \gamma_n] \vdash p[\gamma_1, \ldots, \gamma_n]}\ \texttt{INST\_TYPE}$$

## Simple equality reasoning

We can create various simple derived rules in the usual LCF fashion, such as a one-sided congruence rule:

```
let AP_TERM tm th =
  try MK_COMB(REFL tm,th)
  with Failure _ -> failwith "AP_TERM";;
```

and a symmetry rule to reverse equations:

```
let SYM th =
  let tm = concl th in
  let l,r = dest_eq tm in
  let lth = REFL l in
  EQ_MP (MK_COMB(AP_TERM (rator (rator tm)) th,lth)) lth;;
```

## Logical connectives

Even the logical connectives themselves are defined:

$$\top = (\lambda x.\, x) = (\lambda x.\, x)$$
$$\wedge = \lambda p.\, \lambda q.\, (\lambda f.\, f\ p\ q) = (\lambda f.\, f\ \top\ \top)$$
$$\Rightarrow = \lambda p.\, \lambda q.\, p \wedge q = p$$
$$\forall = \lambda P.\, P = \lambda x.\, \top$$
$$\exists = \lambda P.\, \forall Q.\, (\forall x.\, P(x) \Rightarrow Q) \Rightarrow Q$$
$$\vee = \lambda p.\, \lambda q.\, \forall r.\, (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r$$
$$\bot = \forall P.\, P$$
$$\neg = \lambda t.\, t \Rightarrow \bot$$
$$\exists! = \lambda P.\, \exists P \wedge \forall x.\, \forall y.\, P\ x \wedge P\ y \Rightarrow (x = y)$$

## Building up derived rules

We proceed to get the full HOL Light system by setting up:

- More and more sophisticated derived inference rules, based on earlier ones.
- New types for mathematical structures, defined in terms of earlier structures.

Thus, the whole system is built in a ‘correct by construction’ way and all proofs ultimately reduce to primitives. An early step in the journey is conjunction introduction

$$\frac{\Gamma \vdash p \quad \Delta \vdash q}{\Gamma \cup \Delta \vdash p \wedge q}\ \texttt{CONJ}$$

## Definition of CONJ

. . . which is defined as:

```
let CONJ =
  let f = ‘f:bool->bool->bool‘
  and p = ‘p:bool‘ and q = ‘q:bool‘ in
  let pth =
    let pth = ASSUME p and qth = ASSUME q in
    let th1 = MK_COMB(AP_TERM f (EQT_INTRO pth),EQT_INTRO q
    let th2 = ABS f th1 in
    let th3 = BETA_RULE (AP_THM (AP_THM AND_DEF p) q) in
    EQ_MP (SYM th3) th2 in
  fun th1 th2 ->
    let th = INST [concl th1,p; concl th2,q] pth in
    PROVE_HYP th2 (PROVE_HYP th1 th);;
```

## Some of HOL Light's derived rules

- Simplifier for (conditional, contextual) rewriting.
- Tactic mechanism for mixed forward and backward proofs.
- Tautology checker.
- Automated theorem provers for pure logic, based on tableaux and model elimination.
- Linear arithmetic decision procedures over $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{N}$.
- Differentiator for real functions.
- Generic normalizers for rings and fields
- General quantifier elimination over $\mathbb{C}$
- Gröbner basis algorithm over fields

## A higher-level derived rule

The derived rule `REAL_ARITH` can prove facts of linear arithmetic automatically.

```
REAL_ARITH
  ‘a <= x /\ b <= y /\
   abs(x - y) < abs(x - a) /\
   abs(x - y) < abs(x - b) /\
   (b <= x ==> abs(x - a) <= abs(x - b)) /\
   (a <= y ==> abs(y - b) <= abs(y - a))
   ==> (a = b)‘;;
```

But under the surface, everything is happening by primitive inference (about 50000 such inferences).

## What about the *nonlinear* theory of reals?

The first-order theory of reals is decidable by quantifier elimination:

- 1930: Tarski discovers quantifier elimination procedure for this theory
- 1948: Tarski's algorithm published by RAND
- 1954: Seidenberg publishes simpler algorithm
- 1975: Collins develops and *implements* cylindrical algebraic decomposition (CAD) algorithm
- 1983: Hörmander publishes very simple algorithm based on ideas by Cohen.
- 1990: Vorobjov improves complexity bound to doubly exponential in number of quantifier *alternations*.

## Why is it so little used?

This is an exciting result:

- Illuminates the structure of constructible sets in algebraic geometry
- Gives algorithmic solution for non-trivial questions (e.g. kissing problems in higher dimensions).

Yet there is very little practical use for these methods:

- Theoretical performance is very bad (doubly exponential)
- This is reflected in practical infeasibility even on relatively simple problems.
- Implementation of the best algorithms is complicated, even more if they have to be reliable/certifiable.

## The universal fragment

Consider the case of proving purely universally quantified formulas ('for all $x$, $y$, ...', never 'there exists $z$').

- In principle this seems very restrictive, but it takes in many problems of practical interest.
- Fits naturally with combination methods for 'quantifier-free' decision procedures (satisfiability modulo theories, SMT).
- Permits radically different approaches that can be much more efficient, and a lot easier to certify.

We consider the technique pioneered by Parrilo using sums of squares (SOS).

## Proving nonnegativity of polynomials

We want to prove a polynomial is *positive semidefinite* (PSD):

$$\forall \overline{x}.\, p(\overline{x}) \geqslant 0$$

## Proving nonnegativity of polynomials

We want to prove a polynomial is *positive semidefinite* (PSD):

$$\forall \overline{x}.\, p(\overline{x}) \geqslant 0$$

For a simple example:

$$x^2 - 2x + 1 \geqslant 0$$

## Proving nonnegativity of polynomials

We want to prove a polynomial is *positive semidefinite* (PSD):

$$\forall \overline{x}.\, p(\overline{x}) \geqslant 0$$

For a simple example:

$$x^2 - 2x + 1 = (x-1)^2 \geqslant 0$$

it's a perfect square.

## A more complicated example

$$23x^2 + 6xy + 3y^2 - 20x + 5 \geqslant 0$$

## A more complicated example

$$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geqslant 0$$

## A more complicated example

$$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geqslant 0$$

$$23x^2 + 6xy + 3y^2 - 20x + 5 = \tfrac{1}{23}(23x + 3y - 10)^2 + \tfrac{15}{23}(2y + 1)^2 \geqslant 0$$

## A more complicated example

$$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geqslant 0$$

$$23x^2 + 6xy + 3y^2 - 20x + 5 = \tfrac{1}{23}(23x + 3y - 10)^2 + \tfrac{15}{23}(2y + 1)^2 \geqslant 0$$

We have found *sum of squares* (SOS) decompositions, which suffice to prove nonnegativity.

## From Zeng et al, JSC vol 37, 2004, p83-99

$$w^6 + 2z^2w^3 + x^4 + y^4 + z^4 + 2x^2w + 2x^2z +$$
$$3x^2 + w^2 + 2zw + z^2 + 2z + 2w + 1 \geqslant 0$$

## From Zeng et al, JSC vol 37, 2004, p83-99

$$w^6 + 2z^2w^3 + x^4 + y^4 + z^4 + 2x^2w + 2x^2z +$$
$$3x^2 + w^2 + 2zw + z^2 + 2z + 2w + 1 =$$
$$(y^2)^2 + (x^2 + w + z + 1)^2 + x^2 + (w^3 + z^2)^2 \geqslant 0$$

## Value of SOS techniques

An attractive method providing a very simple certificate for a theorem prover (or person) to verify. But

- Polynomial nonnegativity is a rather special problem, and even then, SOS decomposition may not exist even if the polynomial is PSD
- Not easy to find the SOS decomposition even if it does exist

The solutions to these problems?

- Seek more general 'Positivstellensatz' certificates involving SOS, not just simple SOS decompositions
- Find the Psatz certificates using semidefinite programming.

## The usual Nullstellensatz

Over algebraically closed fields like $\mathbb{C}$ we have a nice simple equivalence.

The polynomial equations $p_1(\overline{x}) = 0, \ldots, p_k(\overline{x}) = 0$ in an algebraically closed field have *no* common solution iff there are polynomials $q_1(\overline{x}), \ldots, q_k(\overline{x})$ such that the following polynomial identity holds:

$$q_1(\overline{x}) \cdot p_1(\overline{x}) + \cdots + q_k(\overline{x}) \cdot p_k(\overline{x}) = 1$$

Thus we can reduce equation-solving to ideal membership and solve it efficiently using Gröbner bases.

## The real Nullstellensatz

In the analogous Nullstellensatz result over $\mathbb{R}$, sums of squares play a central role:

The polynomial equations $p_1(\overline{x}) = 0, \ldots, p_k(\overline{x}) = 0$ in a real closed closed field have *no* common solution iff there are polynomials $q_1(\overline{x})$, $\ldots, q_k(\overline{x})$, $s_1(\overline{x}), \ldots, s_m(\overline{x})$ such that

$$q_1(\overline{x}) \cdot p_1(\overline{x}) + \cdots + q_k(\overline{x}) \cdot p_k(\overline{x}) + s_1(\overline{x})^2 + \cdots + s_m(\overline{x})^2 = -1$$

## The real Positivstellensatz

There are still more general "Positivstellensatz" results about the inconsistency of a set of equations, negated equations, strict and non-strict inequalities.

Can use this to prove any universally quantified formula in the first-order language of reals, e.g. prove

$$\forall a\ b\ c\ x.\ ax^2 + bx + c = 0 \Rightarrow b^2 - 4ac \geqslant 0$$

via the following SOS certificate:

$$b^2 - 4ac = (2ax + b)^2 - 4a(ax^2 + bx + c)$$

## Reduction to semidefinite programming

Can reduce finding SOS decompositions, and PSatz certificates of bounded degree, to *semidefinite programming* (SDP).

SDP is basically optimizing a linear function of parameters while making a matrix linearly parametrized by those parameters PSD.

Can be considered a generalization of linear programming, and similarly is solvable in polynomial time using interior-point algorithms.

There are many efficient tools to solve the problem effectively in practice. I mostly use CSDP.

## Experience and problems

This approach is often much more efficient than competing techniques such as general quantifier elimination.

Lends itself very well to a separation of proof search and LCF-style checking, so fits very well with HOL Light.

Still some awkward numerical problems where the PSD is tight (can become zero) and the rounding to rationals causes loss of PSD-ness.

Available with HOL Light since 2.0 in `Examples/sos.ml`, and seems quite useful. (Includes over-engineered and under-optimized `SOS_CONV`.)

Coq port by Laurent Théry.

## The univariate case

Alternative based on the simple observation that every nonnegative univariate polynomial is a sum of squares of *real* polynomials.

All roots, real or complex, must occur in conjugate pairs. Thus the polynomial is a product of factors

$$(x - [a_k + ib_k])(x - [a_k - ib_k])$$

and so is of the form

$$(q(x) + ir(x))(q(x) - ir(x)) = q(x)^2 + r(x)^2$$

To get an exact rational decomposition, we need a more intricate algorithm, but this is the basic idea.

## Experience of univariate case

Numerical problems can be particularly annoying with some polynomial bound problems in real applications where the coefficients are non-trivial (60-200 bits).

For example, proving $\forall x.\ |x| \leqslant k \Rightarrow |f(x) - p(x)| < \epsilon$ where $p$ is a short approximation to a longer polynomial $f$.

The direct approach is often better than SDP-based methods, for numerical reasons, in such examples.

## Conclusion

Current interactive theorem provers are becoming quite capable and getting applied in formal verification and pure mathematics.

There is currently a ‘gap’ in such systems for nonlinear reasoning over the reals, which despite its theoretical decidability is difficult in practice.

The SOS approach using SDP to find certificates is often more efficient than traditional quantifier elimination, and much better suited to formal certification.

Still some numerical problems; not clear to what extent these would be solved by a high-precision SDP solver.

## Shameless book plug

An introductory survey of many central results in automated reasoning, together with actual code.