Interactive Theorem Proving in Industry

John Harrison

Intel Corporation

16 April 2012

Milner on automation and interaction

I wrote an automatic theorem prover in Swansea for myself and became shattered with the difficulty of doing anything interesting in that direction and I still am. I greatly admired Robinson's resolution principle, a wonderful breakthrough; but in fact the amount of stuff you can prove with fully automatic theorem proving is still very small. So I was always more interested in amplifying human intelligence than I am in artificial intelligence.

Automated theorem proving

The 1970s and 1980s saw intense interest in purely automated theorem proving techniques:

Automated theorem proving

The 1970s and 1980s saw intense interest in purely automated theorem proving techniques:

- Robinson's resolution method and other techniques for first-order logic
- Knuth-Bendix completion for equational logic
- Boyer-Moore style automation of inductive proof
- Shostak and Nelson-Oppen work on cooperating decision procedures, congruence closure

Automated theorem proving

The 1970s and 1980s saw intense interest in purely automated theorem proving techniques:

- Robinson's resolution method and other techniques for first-order logic
- Knuth-Bendix completion for equational logic
- Boyer-Moore style automation of inductive proof
- Shostak and Nelson-Oppen work on cooperating decision procedures, congruence closure

However, when the power of such methods began to plateau, it was hard to make further progress and the field stagnated somewhat.

Interactive theorem proving

Robin Milner was instrumental in emphasizing *interactive* techniques.

Interactive theorem proving

Robin Milner was instrumental in emphasizing *interactive* techniques.

- Milner's original research on Edinburgh LCF spurred an explosion of LCF-stype theorem provers.

Interactive theorem proving

Robin Milner was instrumental in emphasizing *interactive* techniques.

- Milner's original research on Edinburgh LCF spurred an explosion of LCF-stype theorem provers.
- Such systems could be extended by programming without compromising reliability.

Interactive theorem proving

Robin Milner was instrumental in emphasizing *interactive* techniques.

- Milner's original research on Edinburgh LCF spurred an explosion of LCF-stype theorem provers.
- Such systems could be extended by programming without compromising reliability.
- With the development of HOL, the system presented a conservatively constructed mathematical world into which other formalisms could be soundly embedded.

Interactive theorem proving

Robin Milner was instrumental in emphasizing *interactive* techniques.

- Milner's original research on Edinburgh LCF spurred an explosion of LCF-stype theorem provers.
- Such systems could be extended by programming without compromising reliability.
- With the development of HOL, the system presented a conservatively constructed mathematical world into which other formalisms could be soundly embedded.

This led to a renaissance of formalization of all kinds, in pure mathematics and verification.

Further research on automated techniques

However, many important improvements have been made in automation too:

Further research on automated techniques

However, many important improvements have been made in automation too:

- Powerful new decision procedures in algebra and geometry (Gröbner bases, Wu's method).

Further research on automated techniques

However, many important improvements have been made in automation too:

- Powerful new decision procedures in algebra and geometry (Gröbner bases, Wu's method).
- Efficient model checking algorithms for tempoeral logic.

Further research on automated techniques

However, many important improvements have been made in automation too:

- Powerful new decision procedures in algebra and geometry (Gröbner bases, Wu's method).
- Efficient model checking algorithms for tempoeral logic.
- Dazzling efficiency improvements in SAT (and now SMT) solvers makes them surprisingly useful in practice.

Further research on automated techniques

However, many important improvements have been made in automation too:

- Powerful new decision procedures in algebra and geometry (Gröbner bases, Wu's method).
- Efficient model checking algorithms for tempoeral logic.
- Dazzling efficiency improvements in SAT (and now SMT) solvers makes them surprisingly useful in practice.

We are actively trying to combine the power of automated techniques with the generality and reliablity of interactive ones to produce the smoothest and most effective synthesis.

Sound integration of multiple tools

Current applications in both formal verification and the formalization of mathematics most naturally draw on a wide variety of tools.

Sound integration of multiple tools

Current applications in both formal verification and the formalization of mathematics most naturally draw on a wide variety of tools.

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers

Sound integration of multiple tools

Current applications in both formal verification and the formalization of mathematics most naturally draw on a wide variety of tools.

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers
- Some proofs in mathematics use linear programming, nonlinear optimization, computer algebra systems and other more ad hoc algorithms

Sound integration of multiple tools

Current applications in both formal verification and the formalization of mathematics most naturally draw on a wide variety of tools.

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers
- Some proofs in mathematics use linear programming, nonlinear optimization, computer algebra systems and other more ad hoc algorithms
- May want to combine work done in different theorem provers, e.g. ACL2, Coq, HOL, Isabelle.

Sound integration of multiple tools

Current applications in both formal verification and the formalization of mathematics most naturally draw on a wide variety of tools.

- Formal verification uses a wide range of tools including SAT and SMT solvers, model checkers and theorem provers
- Some proofs in mathematics use linear programming, nonlinear optimization, computer algebra systems and other more ad hoc algorithms
- May want to combine work done in different theorem provers, e.g. ACL2, Coq, HOL, Isabelle.

Ideally, we want to be able to retain the soundness guarantees we have grown used to from LCF.

Intel's diverse activities

Intel is best known as a hardware company, and hardware is still the core of the company's business. However this entails much more:

- Microcode
- Firmware
- Protocols
- Software

Intel's diverse activities

Intel is best known as a hardware company, and hardware is still the core of the company's business. However this entails much more:

- Microcode
- Firmware
- Protocols
- Software

If the Intel® Software and Services Group (SSG) were split off as a separate company, it would be in the top 10 software companies worldwide.

Intel's diverse verification problems

This gives rise to a corresponding diversity of verification problems, and of verification solutions.

- Propositional tautology/equivalence checking (FEV)
- Symbolic simulation
- Symbolic trajectory evaluation (STE)
- Temporal logic model checking
- Combined decision procedures (SMT)
- First order automated theorem proving
- Interactive theorem proving

Integrating all these is a challenge!

The Flyspeck project

Hales's Flyspeck project to formally verify his proof of the Kepler conjecture gives rise to similar problems, since it involves many components:

The Flyspeck project

Hales's Flyspeck project to formally verify his proof of the Kepler conjecture gives rise to similar problems, since it involves many components:

- A large amoung of ordinary mathematical formalization (formalized in HOL Light)

The Flyspeck project

Hales's Flyspeck project to formally verify his proof of the Kepler conjecture gives rise to similar problems, since it involves many components:

- A large amoung of ordinary mathematical formalization (formalized in HOL Light)
- Nonlinear optimization (using interval arithmetic and subdivision)

The Flyspeck project

Hales's Flyspeck project to formally verify his proof of the Kepler conjecture gives rise to similar problems, since it involves many components:

- A large amoung of ordinary mathematical formalization (formalized in HOL Light)
- Nonlinear optimization (using interval arithmetic and subdivision)
- Linear programming (using standard LP tools)

The Flyspeck project

Hales's Flyspeck project to formally verify his proof of the Kepler conjecture gives rise to similar problems, since it involves many components:

- A large amoung of ordinary mathematical formalization (formalized in HOL Light)
- Nonlinear optimization (using interval arithmetic and subdivision)
- Linear programming (using standard LP tools)
- Graph enumeration (proved using Isabelle/HOL and run in ML)

The Flyspeck project

Hales's Flyspeck project to formally verify his proof of the Kepler conjecture gives rise to similar problems, since it involves many components:

- A large amoung of ordinary mathematical formalization (formalized in HOL Light)
- Nonlinear optimization (using interval arithmetic and subdivision)
- Linear programming (using standard LP tools)
- Graph enumeration (proved using Isabelle/HOL and run in ML)

This presents a similar integration challenge, since ultimately we would like a unifed and completely formal proof.

Sharing results or sharing proofs?

A key dichotomy is whether we want to simply:

Sharing results or sharing proofs?

A key dichotomy is whether we want to simply:

- Transfer *results*, effectively assuming the soundness of tools

Sharing results or sharing proofs?

A key dichotomy is whether we want to simply:

- Transfer *results*, effectively assuming the soundness of tools
- Transfer *proofs* or other 'certificates' and actually check them in a systematic way.

Sharing results or sharing proofs?

A key dichotomy is whether we want to simply:

- Transfer *results*, effectively assuming the soundness of tools
- Transfer *proofs* or other 'certificates' and actually check them in a systematic way.

The first is general speaking easier and still useful. The latter is more ultimately satisfying and allows us to retain 'LCF-quality' results.

Interfaces between interactive provers

Transferring results:

- hol90 $\to$ Nuprl: Howe and Felty 1997
- ACL2 $\to$ HOL4: Gordon, Hunt, Kaufmann & Reynolds 2006

Transferring proofs:

- HOL4 $\to$ Isabelle/HOL: Skalberg 2006
- HOL Light $\to$ Isabelle/HOL: Obua 2006
- Isabelle/HOL $\to$ HOL Light: McLaughlin 2006
- HOL Light $\to$ Coq: Keller 2009

More comprehensive solutions for exchange between HOL-like provers include work by Hurd et al. (OpenTheory) and Adams (importing into HOL Zero).

Pure logic: SAT

SAT is particularly important nowadays given the power of modern SAT solvers

Pure logic: SAT

SAT is particularly important nowadays given the power of modern SAT solvers

- For *satisfiable* problems it's generally easy to get a satisfying valuation out of a SAT solver and check it relatively efficiently.

Pure logic: SAT

SAT is particularly important nowadays given the power of modern SAT solvers

- For *satisfiable* problems it's generally easy to get a satisfying valuation out of a SAT solver and check it relatively efficiently.
- For *unsatisfiable* problems, some SAT checkers are capable of emitting a resolution proof, and this can be checked.

Pure logic: SAT

SAT is particularly important nowadays given the power of modern SAT solvers

- For *satisfiable* problems it's generally easy to get a satisfying valuation out of a SAT solver and check it relatively efficiently.
- For *unsatisfiable* problems, some SAT checkers are capable of emitting a resolution proof, and this can be checked.

Several reasonably fast solutions, e.g. Weber and Amjad, *Efficiently Checking Propositional Refutations in HOL Theorem Provers*

Pure logic: FOL

In principle, relatively easy: often much faster to check a proof even in a slow prover than to perform the extensive search that led to it. Off-the-shelf provers do create some difficulties:

Pure logic: FOL

In principle, relatively easy: often much faster to check a proof even in a slow prover than to perform the extensive search that led to it. Off-the-shelf provers do create some difficulties:

- Getting a sufficiently explicit proof out of certain provers in the first place.

Pure logic: FOL

In principle, relatively easy: often much faster to check a proof even in a slow prover than to perform the extensive search that led to it. Off-the-shelf provers do create some difficulties:

- Getting a sufficiently explicit proof out of certain provers in the first place.
- Reducing the higher-order polymorphically typed logic to the monomorphic first-order logic supported by most ATPs.

Pure logic: FOL

In principle, relatively easy: often much faster to check a proof even in a slow prover than to perform the extensive search that led to it. Off-the-shelf provers do create some difficulties:

- Getting a sufficiently explicit proof out of certain provers in the first place.
- Reducing the higher-order polymorphically typed logic to the monomorphic first-order logic supported by most ATPs.

Such integrations are currently an active theme, e.g. Isabelle's "Sledgehammer".

Pure logic: QBF

Quantified Boolean formulas are a useful representation for some classes of problem. There have been successful projects to check traces from QBF provers:

Pure logic: QBF

Quantified Boolean formulas are a useful representation for some classes of problem. There have been successful projects to check traces from QBF provers:

- Invalid QBF formulas: Weber 2010

Pure logic: QBF

Quantified Boolean formulas are a useful representation for some classes of problem. There have been successful projects to check traces from QBF provers:

- Invalid QBF formulas: Weber 2010
- Valid QBF formulas: Kuncar 2011, Kumar and Weber 2011

Pure logic: QBF

Quantified Boolean formulas are a useful representation for some classes of problem. There have been successful projects to check traces from QBF provers:

- Invalid QBF formulas: Weber 2010
- Valid QBF formulas: Kuncar 2011, Kumar and Weber 2011

While these work, the process of checking incurs a sometimes dramatic slowdown, and are sensitive to implementation details of the target prover.

Arithmetical theories: linear arithmetic

Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.

Arithmetical theories: linear arithmetic

Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.

Farkas's Lemma, implies that any unsatisfiable set of inequalities has a linear combination that's 'obviously false' like $1 < 0$.

Arithmetical theories: linear arithmetic

Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.

Farkas's Lemma, implies that any unsatisfiable set of inequalities has a linear combination that's 'obviously false' like $1 < 0$.

Obua's initial work and Solovyev's highly optimized refinement is essential for Flyspeck.

Arithmetical theories: linear arithmetic

Generally works quite well for universal formulas over $\mathbb{R}$ or $\mathbb{Q}$.

Farkas's Lemma, implies that any unsatisfiable set of inequalities has a linear combination that's 'obviously false' like $1 < 0$.

Obua's initial work and Solovyev's highly optimized refinement is essential for Flyspeck.

More challenging if we have (i) quantifier alternations, or (ii) non-trivial use of a discrete structures like $\mathbb{Z}$ or $\mathbb{N}$.

Arithmetical theories: algebraically closed fields

Again, the universal theory is easiest, and this coincides with the universal theory of fields or integral domains (when the characteristic is fixed).

Arithmetical theories: algebraically closed fields

Again, the universal theory is easiest, and this coincides with the universal theory of fields or integral domains (when the characteristic is fixed).

Using the Rabinowitsch trick $p \neq 0 \rightarrow \exists y.\ py - 1 = 0$, we just need to refute a conjunction of equations.

Arithmetical theories: algebraically closed fields

Again, the universal theory is easiest, and this coincides with the universal theory of fields or integral domains (when the characteristic is fixed).

Using the Rabinowitsch trick $p \neq 0 \rightarrow \exists y.\ py - 1 = 0$, we just need to refute a conjunction of equations.

Hilbert Nullstellensatz: The polynomial equations $p_1(\overline{x}) = 0, \ldots, p_k(\overline{x}) = 0$ in an algebraically closed field have *no* common solution iff

Arithmetical theories: algebraically closed fields

Again, the universal theory is easiest, and this coincides with the universal theory of fields or integral domains (when the characteristic is fixed).

Using the Rabinowitsch trick $p \neq 0 \rightarrow \exists y.\ py - 1 = 0$, we just need to refute a conjunction of equations.

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

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

Arithmetical theories: algebraically closed fields

Again, the universal theory is easiest, and this coincides with the universal theory of fields or integral domains (when the characteristic is fixed).

Using the Rabinowitsch trick $p \neq 0 \to \exists y.\ py - 1 = 0$, we just need to refute a conjunction of equations.

Hilbert Nullstellensatz: 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, solvable using Gröbner bases.

Arithmetical theories: universal theory of reals (1)

There is an analogous way of certifying universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):

Arithmetical theories: universal theory of reals (1)

There is an analogous way of certifying universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):

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

Arithmetical theories: universal theory of reals (1)

There is an analogous way of certifying universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):

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

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

Arithmetical theories: universal theory of reals (1)

There is an analogous way of certifying universal formulas over $\mathbb{R}$ using the Real Nullstellensatz, which involves sums of squares (SOS):

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 similar but more intricate Positivstellensatz generalizes this to inequalities of all kinds.

Arithmetical theories: universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example

Arithmetical theories: universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example
$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geq 0$

Arithmetical theories: universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example
$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geq 0$

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

Arithmetical theories: universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example
$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geq 0$

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

because

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

Arithmetical theories: universal theory of reals (2)

The appropriate certificates can be found in practice via semidefinite programming (SDP). For example
$23x^2 + 6xy + 3y^2 - 20x + 5 = 5 \cdot (2x - 1)^2 + 3 \cdot (x + y)^2 \geq 0$

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

because

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

However, most standard nonlinear solvers do not return such certificates, and this approach does not obviously generalize to formulas with richer quantifier structure.

Other examples

There has been some research on at least the following:

Other examples

There has been some research on at least the following:

- SMT: seems feasible to combine and generalize methods for SAT and theories.

Other examples

There has been some research on at least the following:

- SMT: seems feasible to combine and generalize methods for SAT and theories.
- Explicit-state or BDD-based symbolic model checking: seems hard to separately certify and emulation is slow.

Other examples

There has been some research on at least the following:

- SMT: seems feasible to combine and generalize methods for SAT and theories.
- Explicit-state or BDD-based symbolic model checking: seems hard to separately certify and emulation is slow.
- Computer algebra: some easy case like factorization, indefinite integrals. Others like definite integrals are much harder.

Other examples

There has been some research on at least the following:

- SMT: seems feasible to combine and generalize methods for SAT and theories.
- Explicit-state or BDD-based symbolic model checking: seems hard to separately certify and emulation is slow.
- Computer algebra: some easy case like factorization, indefinite integrals. Others like definite integrals are much harder.

Major research challenge: which algorithms lend themselves to this kind of efficient checking? Which ones seem essentially not to? Some analogies with the class NP.

Fully integrated automation?

Suppose we have many efficient decision procedures implemented by external tools. How can we put them together?

Fully integrated automation?

Suppose we have many efficient decision procedures implemented by external tools. How can we put them together?

Effectively combination methods like Nelson-Oppen and Shostak solve this problem for quantifier-free theories.

Fully integrated automation?

Suppose we have many efficient decision procedures implemented by external tools. How can we put them together?

Effectively combination methods like Nelson-Oppen and Shostak solve this problem for quantifier-free theories.

Even mild extensions with quantifiers rapidly become undecidable, such as linear integer arithmetic with one function symbol, when we can characterize squaring:

$$(\forall n. f(-n) = f(n)) \wedge f(0) = 0 \wedge (\forall n. 0 \leq n \Rightarrow f(n+1) = f(n) + n + n + 1)$$

and then multiplication by $m = n \cdot p \Leftrightarrow (n+p)^2 = n^2 + p^2 + 2m$

Quantifiers + theories

At present, we still seem to need human-driven interactive proof to formulate lemmas that can be solved by automated tools and tie them together.

Quantifiers + theories

At present, we still seem to need human-driven interactive proof to formulate lemmas that can be solved by automated tools and tie them together.

One of the primary research problems in automated theorem proving is to find a practically effective combination of quantifier and theory reasoning.

Quantifiers + theories

At present, we still seem to need human-driven interactive proof to formulate lemmas that can be solved by automated tools and tie them together.

One of the primary research problems in automated theorem proving is to find a practically effective combination of quantifier and theory reasoning.

- First-order provers are adding theory reasoning (SPASS+T)

Quantifiers + theories

At present, we still seem to need human-driven interactive proof to formulate lemmas that can be solved by automated tools and tie them together.

One of the primary research problems in automated theorem proving is to find a practically effective combination of quantifier and theory reasoning.

- First-order provers are adding theory reasoning (SPASS+T)
- SMT solvers are improving their ability to instantiate quantifiers

Quantifiers + theories

At present, we still seem to need human-driven interactive proof to formulate lemmas that can be solved by automated tools and tie them together.

One of the primary research problems in automated theorem proving is to find a practically effective combination of quantifier and theory reasoning.

- First-order provers are adding theory reasoning (SPASS+T)
- SMT solvers are improving their ability to instantiate quantifiers

Can sometimes exploit types to instantiate quantifiers systematically, and other heuristics often seem to work well in practice.

Conclusions

- There is a real need for combining different proof tools, for applications both in formal verification and pure mathematics

Conclusions

- There is a real need for combining different proof tools, for applications both in formal verification and pure mathematics
- Effective exchange and checking of proofs between tools seems to be the best way of maintaining the 'LCF advantage'.

Conclusions

- There is a real need for combining different proof tools, for applications both in formal verification and pure mathematics
- Effective exchange and checking of proofs between tools seems to be the best way of maintaining the 'LCF advantage'.
- Several significant problems still seem hard to treat effectively via a certification, including model checking state enumeration and full quantifier elimination or general nonlinear optimization.

Conclusions

- There is a real need for combining different proof tools, for applications both in formal verification and pure mathematics
- Effective exchange and checking of proofs between tools seems to be the best way of maintaining the 'LCF advantage'.
- Several significant problems still seem hard to treat effectively via a certification, including model checking state enumeration and full quantifier elimination or general nonlinear optimization.
- The final challenge will probably lie in the effective combination of a variety of certified techniques, which broadly involves the combination of quantifier and theory reasoning.