A Sketch of the History and State of Automated Deduction

David A. Plaisted

October 19, 2015

Introduction

- Ed Reingold influenced me to do algorithms research early in my career, and this led to some good work. However, my main research area now is automatic theorem proving, so I will discuss this instead.
- This talk will be a survey touching on some aspects of the state of the field of automated deduction at present. The content of the talk will be somewhat more general than the original title indicated; thus the change in title. This talk should be about 20 minutes long. I'm planning to put these notes on the internet soon. There is a lot to cover in 20 minutes, so it is best to leave most questions until the end.

Proofs in Computer Science

- There is a need for formal proofs in computer science, for example, to prove the correctness of software, or to prove theorems in a theoretical paper.
- There is also a need for proofs in other areas such as mathematics.
- It can be helpful to have programs that check proofs or even help to find the proofs.

Automated Deduction

- This talk emphasizes systems that do a lot of proof search on their own, either to find the whole proof automatically or large parts of it, without the user specifying each application of each rule of inference.
- There are also systems that permit the user to construct a proof and check that the rules of inference are properly applied.

History of Automated Deduction

Our current state of knowledge in automated deduction came to us by a long and complicated process of intellectual development. Some excellent discussions of this are from Davis[Dav83] and Bibel[Bib07].

- Gottfried Wilhelm Leibniz (b. 1646, d. 1716) was a German philosopher, mathematician, and logician who
  - invented the differential and integral calculus independently of Newton and who also
  - had the concept of mechanizing reason (see Davis[Dav83]).
  - His work influenced many later researchers.
- Friedrich Ludwig Gottlob Frege (b. 1848, d. 1925) was a German mathematician, logician, and philosopher who worked at the University of Jena.
  - Frege essentially reconceived the discipline of logic by constructing a formal system which, in effect, constituted the first predicate calculus.
  - In this formal system, Frege developed an analysis of quantified statements and formalized the notion of a proof in terms that are still accepted today.
  - Frege then demonstrated that one could use his system to resolve theoretical mathematical statements in terms of simpler logical and mathematical notions.

[Stanford Encyclopedia of Philosophy]

- David Hilbert (1862 - 1943) was a German mathematician. Among many other contributions, he wanted to provide a secure foundation for mathematics including a formalization of mathematics and an algorithm for deciding the truth or falsity of any mathematical statement. This is known as Hilberts program. (from Wikipedia)
  - Gödel showed that Hilbert's program was impossible, in its obvious interpretation.
  - The ATP community has inherited Hilberts program to some extent, in attempting to prove and decide what can be proved and decided.

First-Order Logic

This talk will emphasize first-order logic.

- In this system, there are predicates, functions, variables, universal and existential quantifiers, and Boolean connectives.
- This system is very expressive, but other systems involving higher order logic are even more expressive.
- An example of a first-order formula is

$$\forall x(P(x) \supset \exists y Q(f(x), y)).$$

Mechanizing Reasoning

- We are interested in computer programs that realize Leibniz's dream of mechanizing logical reasoning.
- Computers are ideally suited to find and check proofs because they are fast and accurate, but lack insight and creativity.
- The first published implementation of a fully general theorem prover for first-order logic was that of Prawitz, discussed in Bibel[Bib07].

History of Mechanizing Reasoning

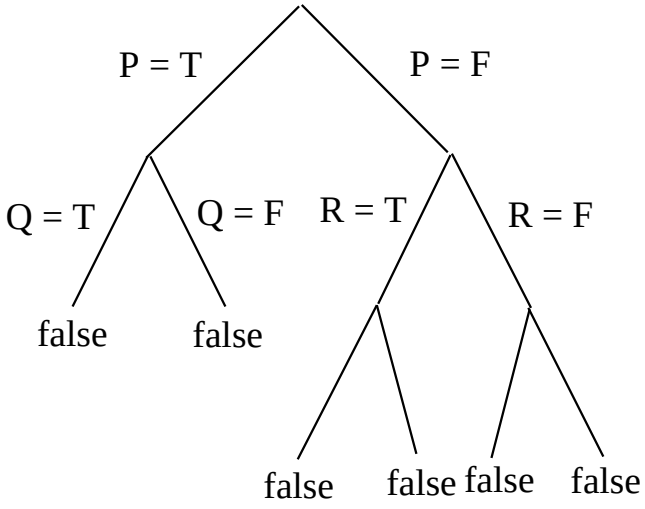
- Davis and Putnam's method 1960 used ground resolution as part of an approach to proving formulas in first-order logic
  - Ground resolution takes two propositional clauses and produces another clause:

$$\frac{\neg P \vee Q \vee R \quad \neg R \vee S}{\neg P \vee Q \vee S}$$

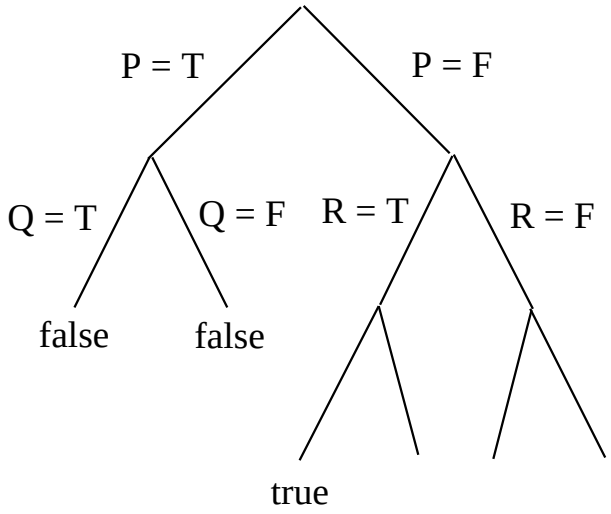
  - Davis and Putnam's method performs a sequence of such ground resolutions to test a Boolean formula for satisfiability.
- DPLL 1962 tests a Boolean formula for satisfiability by a backtracking search instead of using ground resolution as in Davis and Putnam's method.
  - DPLL is much faster on propositional formulas than Davis and Putnam's method.
  - Example of a Boolean formula:

$$(P \wedge Q) \supset (Q \vee P \vee R)$$

  - DPLL involves a systematic search for a model (satisfying truth assignment) with backtracking
  - Here is a tree of possible assignments of a few predicate symbols to illustrate how it is searched and what happens at failure nodes.



Unsatisfiable Formula



Satisfiable Formula

- At failure or success nodes the formula simplifies to true or false. One true: sat, all false: unsat.
- Very fast today though NP complete

Efficiency of Sat Solving

- Problems in other domains are often translated into propositional calculus so that a sat solver can be used on them

> The announcement of the Special issue of the Journal of Automated Reasoning for SAT 2005 [GW04] stated, "Systematic methods can now routinely solve verification problems with thousands or tens of thousands of variables, whilst local search methods can solve hard random 3SAT problems with millions of variables."

NP completeness and satisfiability threshold

- When the ratio of clauses to variables is high, DPLL is likely to backtrack soon so that the search tree is small.
- When the ratio is low, DPLL is likely to find a model quickly.
- The hard problems tend to be those in the middle.
- Let α be the ratio of clauses to variables.
    1. For the case of 3-SAT, when α is greater than a number that has been experimentally determined to be approximately 4.27, then almost all random 3-SAT formulas are unsatisfiable, that is, the fraction of unsatisfiable formulas tends to 1.
    2. The opposite is true when $\alpha < 4.27$.
    3. Analogous phenomena have been observed for k-SAT with $k > 3$,and the experimentally determined threshold point increases with k[LMK05].

Satisfiability modulo Theories (SMT); DPLL Extended to SMT

- From Wikipedia:

1. In computer science and mathematical logic, the satisfiability modulo theories (SMT) problem is a decision problem for logical formulas with respect to combinations of background theories expressed in classical first-order logic with equality.
2. Examples of theories typically used in computer science are the theory of real numbers, the theory of integers, and the theories of various data structures such as lists, arrays, bit vectors and so on.

- This is from the announcement for the Fifth International SAT/SMT Summer School, Stanford, CA, July 15-17, 2015:

> Satisfiability (SAT) and Satisfiability Modulo Theories (SMT) solvers have become the engines powering numerous applications in computer science and beyond, including automated verification, artificial intelligence, program synthesis, security, product configuration, and many more. ...

- Z3 is a high-performance SMT theorem prover being developed at Microsoft Research.
  1. Z3 supports linear real and integer arithmetic, fixed-size bit-vectors, extensional arrays, uninterpreted functions, and quantifiers.
  2. Z3 is integrated with a number of program analysis, testing, and verification tools from Microsoft Research.
  3. These include: VCC, Spec#, Boogie, Pex, Yogi, Vigilante, SLAM, F7, F*, SAGE, VS3, FORMULA, and HAVOC.

Resolution and First-Order Logic

- Robinson resolution 1965
- Applies resolution directly to first-order clauses instead of translating them to Boolean formulas first as in Davis and Putnam's method
- Resolution operates on two first-order clauses to produce another clause
- Example of a first-order clause:

$$\neg P(x) \vee \neg Q(y) \vee R(f(x), g(y))$$

Effects of Resolution

- Resolution led to a dramatic increase in power of general first-order logic theorem provers
- Leading provers now Vampire E Spass all basically resolution provers
- Vampire by Andrei Voronkov wins most of the classes now in CASC, CADE's annual prover competition

Shift in the Field

- Resolution has dominated first-order theorem proving since 1965
- The field has been handicapped for a long time by some inefficiencies of resolution
- Propositional methods similar to DPLL are now coming to the fore in first-order automatic theorem proving

Instance-Based Methods

- Instance-based methods take a set of first-order clauses, generate a set of *instances* obtained by replacing the variables by terms, and then apply a DPLL-style method to these instances.
- Instance-based methods essentially are methods of applying DPLL to first-order logic without translating first to Boolean formulas.

- Some examples (CLIN, Inst-Gen, DCTP, OSHL)
- Much faster than resolution on certain kinds of problems
- My work either started or revived this research area
- Vampire has had instance based methods built into it (InstGen) for some time

AVATAR

- Very recent development: Avatar (and extension to SMT?) built into Vampire
- Andrei Voronkov talk in Berlin in August 2015
- A first-order clause is *ground* if it has no variables; such clauses are essentially propositional so that DPLL can be directly applied to them.
- A first-order clause is *splittable* if it can be partitioned into subclauses that do not share variables. Note that all ground clauses are splittable.
- In AVATAR, ground clauses and splittable clauses produced by resolution are handled by a DPLL style search that breaks the theorem into parts to prove separately, and co-ordinates with the resolution part of the prover

Performance of AVATAR

- Appears to be a breakthrough in the field
- Found proofs for hundreds of problems never before solved by any automatic first-order provers.
- This is unusual, to get so many new problems.
- Open problem in semigroup theory solved, over 10,000 cases organized and solved one by one by the prover

Implications

- Shows that progress can still be made in pure first-order logic theorem proving
- I believe that there are more advances in the field yet to come.
- For example SMT can be used in Avatar instead of DPLL on the propositional parts of the clauses
- This should lead to another quantum increase in the power of the prover
- Will have effects also in higher order logics and formalizing mathematics

Isabelle

Isabelle is a generic proof assistant which is capable of using higher-order logics that are more expressive than first-order logic.

- It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus.
- The main application is the formalization of mathematical proofs and in particular formal verification.
- Isabelle was originally developed at the University of Cambridge and Technische Universität München, but now includes numerous contributions from institutions and individuals worldwide.

Tools

For proofs, Isabelle incorporates some tools to improve the user's productivity.

- Isabelle's classical reasoner can perform long chains of reasoning steps to prove formulas.
- The simplifier can reason with and about equations.
- Linear arithmetic facts are proved automatically, various algebraic decision procedures are provided.
- External first-order provers can be invoked through sledgehammer.

Library

Isabelle comes with a large theory library of formally verified mathematics, including

- 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).
- Also provided are numerous examples arising from research into formal verification.

Sledgehammer

- Sledgehammer uses first-order provers to help prove higher order theorems in Isabelle
- Get hints from first order provers, which lemmas are needed for the proof, then re-do the proof in Isabelle

Applications of Sledgehammer

- Christian Urban (winner of Skolem 2005) using Sledgehammer and Isabelle checking a paper by a well–known computer scientist, found and fixed bugs faster than the author could.
- Talked to Urban in Berlin this summer
- He said the saying in the Isabelle community is, Plan A: Sledgehammer Plan B: Think
- If sledgehammer finds the proof you don't have to worry so much about how to prove a lemma or even what it means
- In this way first order provers are impacting higher order provers like Isabelle too
- But first-order provers are becoming more powerful now, and even moreso with Avatar
- Thus interactive proof finding tools such as Isabelle are becoming more and more powerful and potentially more useful to researchers writing papers

Hereditarily Finite Sets in Isabelle

The need

- All systems including Isabelle have restrictions to avoid Russell's paradox
- These can be difficult to comply with in formalizing proofs
- This has made it difficult in the past for Isabelle to formalize some fairly simple theories such as automata theory

The solution

- Larry Paulson presented a paper about this in Berlin this summer
- Hereditarily finite sets; recursive definition:
  1. The empty set is hereditarily finite.
  2. A finite set of hereditarily finite sets is hereditarily finite.
  3. The hereditarily finite sets are the smallest collection of sets having these properties.
- These are essentially sets that can be written using only braces { } in a finite number of symbols, such as {{},{{},{{}}}}.
- Such sets should be enough to express automata of various kinds in Isabelle and the data structures they operate on

Applications

- This should make Isabelle more useful for formalizing reasoning in computer science and other areas, making it a more useful tool for finding and checking proofs.
- Classical automata theory proofs such as minimizing finite automata, nondeterministic finite automata to deterministic conversion, et cetera, are now easily provable on Isabelle

Verifying Mathematical Proofs

[From Formally Verified Mathematics, Jeremy Avigad, John Harrison, Communications of the ACM, Vol. 57 No. 4, Pages 66-75].

This section concentrates on the interactive use of computers to check proofs, rather than using the computer to find the proof automatically.

Errors in Proofs

A book written by Lecat in 1935 included 130 pages of errors made by major mathematicians up to 1900, and even mathematicians of the stature of J.E. Littlewood have published faulty proofs.

Mistakes in Four-Color Theorem Proof

- The first purported proof of the four-color theorem in 1879 stood for a decade before a flaw was pointed out.

Mistsakes in Wiles' Proof of Fermat's Last Theorem

- Referees reviewing Andrew Wiles's first proof of Fermat's Last Theorem found a mistake, and it took Wiles and a former student, Richard Taylor, close to a year to find a way to circumvent it.

Mistake in Classification of Finite Simple Groups Proof

- Daniel Gorenstein announced, in 1983, that the classification of finite simple groups had been completed, unaware there was a gap in the treatment of the class of "quasithin" groups.
- The gap was not filled until 2001, and doing so required a 1,221-page proof by Michael Aschbacher and Stephen Smith.

Proofs involving Extensive Computer Enumerations

Some computer proofs require exhaustive enumerations of many cases. Such complicated proofs are prone to error and need computer verification.

Four color Theorem

Kenneth Appel's and Wolfgang Hakken's 1976 proof of the four-color theorem relied on an exhaustive computer enumeration of combinatorial configurations. Subsequent proofs, though more efficient, have this same character.

Kepler Conjecture

But this feature took on dramatic proportions with Thomas Hales's 1998 proof of the Kepler conjecture, stating that no packing of spheres in 3D space has higher density than the natural face-centered cubic packing commonly used to stack oranges, cannonballs, and such.

- Hales, working with Samuel Ferguson, arrived at a proof in 1998 consisting of 300 pages of mathematics and calculations performed by approximately 40,000 lines of computer code.
- As part of the peer-review process, a panel of 12 referees appointed by the Annals of Mathematics studied the proof for four full years, finally returning with the verdict that they were "99 percent certain" of the correctness, ...

Major Verifications

Feit-Thompson Theorem Verified by Computer

Interactive theorem proving reached a major landmark on September 20, 2012, when Georges Gonthier announced he and a group of researchers under his direction had completed a verification of the Feit-Thompson theorem. ...

- The Feit-Thompson theorem, sometimes called the odd-order theorem, says every finite group of odd order is solvable; equivalently, that the finite simple groups of odd order are exactly the cyclic groups of prime order. ...
- The original proof by Walter Feit and John Thompson, published in 1963, filled 255 journal pages.

Verification of Feit-Thompson in Coq

Gonthier launched the project in 2006 ...

- Because Coq is based on a constructive logic, Gonthier had to reorganize the proof in such a way every theorem has a direct computational interpretation.
- The resulting proof has approximately 150,000 lines of "code," or formal proof scripts, including 4,000 definitions and 13,000 lemmas and theorems. ...

Flyspeck Theorem

Hales's Flyspeck project is another ambitious formalization effort. In response to the outcome of the referee process at the Annals, Hales decided to formally verify a proof of the Kepler conjecture.

Computer Verification of the Flyspeck Theorem

The proof involves three essential uses of computation:

- enumerating a class of combinatorial structures called "tame hypermaps";
- using linear-programming methods to establish bounds on a large number of systems of linear constraints; and
- using interval methods to verify approximately 1,000 nonlinear inequalities that arise in the proof.
- All this is in addition to the textual "paper" proof, which in and of itself is quite long ...
- After a substantial effort by a large, geographically distributed team, the project is nearing completion. In fact it has now been completed, using a combination of Isabelle and HOL Light.

Difficulty of Proof Verifications

Despite recent advances, however, the technology is not quite ready for prime time.

- The formal methods can be difficult to learn and time consuming to apply.
- Hales suspects the Flyspeck project has already exceeded his initial estimate of 20 person-years to completion.

Factor of Four Increase

- One way to quantify the difficulty of a formalization is to compare its length to the length of the original proof, a ratio known as the "de Bruijn factor."
- Freek Wiedijk carried out a short study and, in the three examples considered, the ratio hovers around a factor of four, ...

Help from Automated Deduction

As automatic theorem provers become more powerful, such complicated verifications should become easier as computers take over more of the task of automatically proving larger and larger proof steps.

The Robbins Problem

- An example of a substantial proof found completely automatically
- McCunes proof of the Robbins conjecture in October 1996.
- The successful search took about 8 days on an RS/6000 processor and used about 30 megabytes of memory.
- In 1933, E. V. Huntington presented the following basis for Boolean algebra:
  - $x + y = y + x$. [commutativity]
  - $(x + y) + z = x + (y + z)$. [associativity]
  - $n(n(x) + y) + n(n(x) + n(y)) = x$. [Huntington equation]

- Shortly thereafter, Herbert Robbins conjectured that the Huntington equation can be replaced with a simpler one:

  – n(n(x + y) + n(x + n(y))) = x. [Robbins equation]

- Robbins and Huntington could not find a proof, and the problem was later studied by Tarski and his students:

  (from `https://www.cs.unm.edu/~mccune/papers/robbins/`)

Other Topics

- complexity of first-order proving; partial decidability
- topology and accumulation points

References

[Bib07] Wolfgang Bibel. Early history and perspectives of automated deduction. In *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007, Proceedings*, pages 2–18, 2007.

[Dav83] M. Davis. The prehistory and early history of automated deduction. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning 1: Classical Papers on Computational Logic 1957-1966*, pages 1–28. Springer, Berlin, Heidelberg, 1983.

[GW04] Enrico Giunchiglia and Toby Walsh. SAT 2005, January 2004.

[LMK05] Michele Zito Lefteris M. Kirousis, Yannis C. Stamatiou. The satisfiability threshold conjecture:techniques behind upper bound improvements. Technical report, Santa Fe, New Mexico, USA, 2005.