Automated theorem proving

Automated theorem proving (also known as **ATP** or **automated deduction**) is a subfield of automated reasoning and mathematical logic dealing with proving mathematical theorems by computer programs. Automated reasoning over mathematical proof was a major motivating factor for the development of computer science.

Logical foundations

While the roots of formalised logic go back to Aristotle, the end of the 19th and early 20th centuries saw the development of modern logic and formalised mathematics. Frege's *Begriffsschrift* (1879) introduced both a complete propositional calculus and what is essentially modern predicate logic.[1] His *Foundations of Arithmetic*, published in 1884,[2] expressed (parts of) mathematics in formal logic. This approach was continued by Russell and Whitehead in their influential *Principia Mathematica*, first published 1910–1913,[3] and with a revised second edition in 1927.[4] Russell and Whitehead thought they could derive all mathematical truth using axioms and inference rules of formal logic, in principle opening up the process to automation. In 1920, Thoralf Skolem simplified a previous result by Leopold Löwenheim, leading to the Löwenheim–Skolem theorem and, in 1930, to the notion of a Herbrand universe and a Herbrand interpretation that allowed (un)satisfiability of first-order formulas (and hence the validity of a theorem) to be reduced to (potentially infinitely many) propositional satisfiability problems.[5]

In 1929, Mojżesz Presburger showed that the first-order theory of the natural numbers with addition and equality (now called Presburger arithmetic in his honor) is decidable and gave an algorithm that could determine if a given sentence in the language was true or false.[6][7]

However, shortly after this positive result, Kurt Gödel published *On Formally Undecidable Propositions of Principia Mathematica and Related Systems* (1931), showing that in any sufficiently strong axiomatic system there are true statements that cannot be proved in the system. This topic was further developed in the 1930s by Alonzo Church and Alan Turing, who on the one hand gave two independent but equivalent definitions of computability, and on the other gave concrete examples of undecidable questions.

First implementations

Shortly after World War II, the first general-purpose computers became available. In 1954, Martin Davis programmed Presburger's algorithm for a JOHNNIAC vacuum-tube computer at the Institute for Advanced Study in Princeton, New Jersey. According to Davis, "Its great triumph was to prove that the sum of two even numbers is even".[7][8] More ambitious was the Logic Theorist in 1956, a deduction system for the propositional logic of the *Principia Mathematica*, developed by Allen Newell, Herbert A. Simon and J. C. Shaw. Also running on a JOHNNIAC, the Logic Theorist constructed proofs from a small set of propositional axioms and three deduction rules: modus ponens, (propositional) variable substitution, and the replacement of formulas by their definition. The system used heuristic guidance, and managed to prove 38 of the first 52 theorems of the *Principia*.[7]

The "heuristic" approach of the Logic Theorist tried to emulate human mathematicians, and could not guarantee that a proof could be found for every valid theorem even in principle. In contrast, other, more systematic algorithms achieved, at least theoretically, completeness for first-order logic. Initial approaches relied on the results of Herbrand and Skolem to convert a first-order formula into successively larger sets of propositional formulae by instantiating variables with terms from the Herbrand universe. The propositional formulas could then be checked for unsatisfiability using a number of methods. Gilmore's program used conversion to disjunctive normal form, a form in which the satisfiability of a formula is obvious.[7][9]

Decidability of the problem

Depending on the underlying logic, the problem of deciding the validity of a formula varies from trivial to impossible. For the common case of propositional logic, the problem is decidable but co-NP-complete, and hence only exponential-time algorithms are believed to exist for general proof tasks. For a first-order predicate calculus, Gödel's completeness theorem states that the theorems (provable statements) are exactly the semantically valid well-formed formulas, so the valid formulas are computably enumerable: given unbounded resources, any valid formula can eventually be proven. However, *invalid* formulas (those that are *not* entailed by a given theory), cannot always be recognized.

The above applies to first-order theories, such as Peano arithmetic. However, for a specific model that may be described by a first-order theory, some statements may be true but undecidable in the theory used to describe the model. For example, by Gödel's incompleteness theorem, we know that any consistent theory whose axioms are true for the natural numbers cannot prove all first-order statements true for the natural numbers, even if the list of axioms is allowed to be infinite enumerable. It follows that an automated theorem prover will fail to terminate while searching for a proof precisely when the statement being investigated is undecidable in the theory being used, even if it is true in the model of interest. Despite this theoretical limit, in practice, theorem provers can solve many hard problems, even in models that are not fully described by any first-order theory (such as the integers).

Related problems

A simpler, but related, problem is *proof verification*, where an existing proof for a theorem is certified valid. For this, it is generally required that each individual proof step can be verified by a primitive recursive function or program, and hence the problem is always decidable.

Since the proofs generated by automated theorem provers are typically very large, the problem of proof compression is crucial, and various techniques aiming at making the prover's output smaller, and consequently more easily understandable and checkable, have been developed.

Proof assistants require a human user to give hints to the system. Depending on the degree of automation, the prover can essentially be reduced to a proof checker, with the user providing the proof in a formal way, or significant proof tasks can be performed automatically. Interactive provers are used for a variety of tasks, but even fully automatic systems have proved a number of interesting and hard theorems,

including at least one that has eluded human mathematicians for a long time, namely the Robbins conjecture.[10][11] However, these successes are sporadic, and work on hard problems usually requires a proficient user.

Another distinction is sometimes drawn between theorem proving and other techniques, where a process is considered to be theorem proving if it consists of a traditional proof, starting with axioms and producing new inference steps using rules of inference. Other techniques would include model checking, which, in the simplest case, involves brute-force enumeration of many possible states (although the actual implementation of model checkers requires much cleverness, and does not simply reduce to brute force).

There are hybrid theorem proving systems that use model checking as an inference rule. There are also programs that were written to prove a particular theorem, with a (usually informal) proof that if the program finishes with a certain result, then the theorem is true. A good example of this was the machine-aided proof of the four color theorem, which was very controversial as the first claimed mathematical proof that was essentially impossible to verify by humans due to the enormous size of the program's calculation (such proofs are called non-surveyable proofs). Another example of a program-assisted proof is the one that shows that the game of Connect Four can always be won by the first player.

Applications

Commercial use of automated theorem proving is mostly concentrated in integrated circuit design and verification. Since the Pentium FDIV bug, the complicated floating point units of modern microprocessors have been designed with extra scrutiny. AMD, Intel and others use automated theorem proving to verify that division and other operations are correctly implemented in their processors.[12]

Other uses of theorem provers include program synthesis, constructing programs that satisfy a formal specification.[13] Automated theorem provers have been integrated with proof assistants, including Isabelle/HOL.[14]

Applications of theorem provers are also found in natural language processing and formal semantics, where they are used to analyze discourse representations.[15][16]

First-order theorem proving

In the late 1960s agencies funding research in automated deduction began to emphasize the need for practical applications. One of the first fruitful areas was that of program verification whereby first-order theorem provers were applied to the problem of verifying the correctness of computer programs in languages such as Pascal, Ada, etc. Notable among early program verification systems was the Stanford Pascal Verifier developed by David Luckham at Stanford University.[17][18][19] This was based on the Stanford Resolution Prover also developed at Stanford using John Alan Robinson's resolution principle. This was the first automated deduction system to demonstrate an ability to solve mathematical problems that were announced in the *Notices of the American Mathematical Society* before solutions were formally published.

First-order theorem proving is one of the most mature subfields of automated theorem proving. The logic is expressive enough to allow the specification of arbitrary problems, often in a reasonably natural and intuitive way. On the other hand, it is still semi-decidable, and a number of sound and complete calculi have been developed, enabling *fully* automated systems.[20] More expressive logics, such as higher-order logics, allow the convenient expression of a wider range of problems than first-order logic, but theorem proving for these logics is less well developed.[21][22]

Relationship with SMT

There is substantial overlap between first-order automated theorem provers and SMT solvers. Generally, automated theorem provers focus on supporting full first-order logic with quantifiers, whereas SMT solvers focus more on supporting various theories (interpreted predicate symbols). ATPs excel at problems with lots of quantifiers, whereas SMT solvers do well on large problems without quantifiers.[23] The line is blurry enough that some ATPs participate in SMT-COMP, while some SMT solvers participate in CASC.[24]

Benchmarks, competitions, and sources

The quality of implemented systems has benefited from the existence of a large library of standard benchmark examples—the Thousands of Problems for Theorem Provers (TPTP) Problem Library[25]—as well as from the CADE ATP System Competition (CASC), a yearly competition of first-order systems for many important classes of first-order problems.

Some important systems (all have won at least one CASC competition division) are listed below.

- E is a high-performance prover for full first-order logic, but built on a purely equational calculus, originally developed in the automated reasoning group of Technical University of Munich under the direction of Wolfgang Bibel, and now at Baden-Württemberg Cooperative State University in Stuttgart.
- Otter, developed at the Argonne National Laboratory, is based on first-order resolution and paramodulation. Otter has since been replaced by Prover9, which is paired with Mace4.
- SETHEO is a high-performance system based on the goal-directed model elimination calculus, originally developed by a team under direction of Wolfgang Bibel. E and SETHEO have been combined (with other systems) in the composite theorem prover E-SETHEO.
- Vampire was originally developed and implemented at Manchester University by Andrei Voronkov and Kryštof Hoder. It is now developed by a growing international team. It has won the FOF division (among other divisions) at the CADE ATP System Competition regularly since 2001.[26]
- Waldmeister is a specialized system for unit-equational first-order logic developed by Arnim Buch and Thomas Hillenbrand. It won the CASC UEQ division for fourteen consecutive years (1997–2010).
- SPASS is a first-order logic theorem prover with equality. This is developed by the research group Automation of Logic, Max Planck Institute for Computer Science.

The Theorem Prover Museum[27] is an initiative to conserve the sources of theorem prover systems for future analysis, since they are important cultural/scientific artefacts. It has the sources of many of the systems mentioned above.

Popular techniques

- First-order resolution with unification
- Model elimination
- Method of analytic tableaux
- Superposition and term rewriting
- Model checking
- Mathematical induction[28]
- Binary decision diagrams
- DPLL
- Higher-order unification

- Quantifier elimination[29]

Software systems

Comparison

Name	License type	Web service	Library	Standalone	Last update (YYYY-mm-dd format)
ACL2	3-clause BSD	No	No	Yes	May 2019
Prover9/Otter	Public Domain	Via System on TPTP	Yes	No	2009
Jape	GPLv2	Yes	Yes	No	May 15, 2015
PVS	GPLv2	No	Yes	No	January 14, 2013
EQP	?	No	Yes	No	May 2009
PhoX	?	No	Yes	No	September 28, 2017
E	GPL	Via System on TPTP	No	Yes	July 4, 2017
SNARK	Mozilla Public License 1.1	No	Yes	No	2012
Vampire	Vampire License (https://vprover.github.io/licence.html)	Via System on TPTP	Yes	Yes	December 14, 2017
Theorem Proving System (TPS)	TPS Distribution Agreement (http://gtps.math.cmu.edu/cgi-bin/tpsdist.pl)	No	Yes	No	February 4, 2012
SPASS	FreeBSD license	Yes	Yes	Yes	November 2005
IsaPlanner	GPL	No	Yes	Yes	2007
KeY	GPL	Yes	Yes	Yes	October 11, 2017
Z3 Theorem Prover	MIT License	Yes	Yes	Yes	November 19, 2019

Free software

- Alt-Ergo
- Automath
- CVC
- E
- IsaPlanner
- LCF
- Mizar
- NuPRL
- Paradox
- Prover9

- PVS
- SPARK (programming language)
- Twelf
- Z3 Theorem Prover

Proprietary software

- CARINE
- Wolfram Mathematica
- ResearchCyc

See also

- Curry–Howard correspondence
- Symbolic computation
- Ramanujan machine
- Computer-aided proof
- Formal verification
- Logic programming
- Proof checking
- Model checking
- Proof complexity
- Computer algebra system
- Program analysis (computer science)
- General Problem Solver
- Metamath language for formalized mathematics
- De Bruijn factor

Notes

1. Frege, Gottlob (1879). *Begriffsschrift* (http://gallica.bnf.fr/ark:/12148/bpt6k65658c). Verlag Louis Neuert.
2. Frege, Gottlob (1884). *Die Grundlagen der Arithmetik* (https://web.archive.org/web/20070926172317/http://www.ac-nancy-metz.fr/enseign/philo/textesph/Frege.pdf) (PDF). Breslau: Wilhelm Kobner. Archived from the original (http://www.ac-nancy-metz.fr/enseign/philo/textesph/Frege.pdf) (PDF) on 2007-09-26. Retrieved 2012-09-02.
3. Russell, Bertrand; Whitehead, Alfred North (1910–1913). *Principia Mathematica* (https://archive.org/details/cu31924001575244) (1st ed.). Cambridge University Press.
4. Russell, Bertrand; Whitehead, Alfred North (1927). *Principia Mathematica* (https://archive.org/details/in.ernet.dli.2015.221192) (2nd ed.). Cambridge University Press.
5. Herbrand, J. (1930). *Recherches sur la théorie de la démonstration* (https://eudml.org/doc/192791) (PhD) (in French). University of Paris.
6. Presburger, Mojżesz (1929). "Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt". *Comptes Rendus du I Congrès de Mathématiciens des Pays Slaves*. Warszawa: 92–101.

7. Davis, Martin (2001). "The Early History of Automated Deduction" (https://web.archive.org/web/20120728092819/http://www.cs.nyu.edu/cs/faculty/davism/early.ps). *Robinson & Voronkov 2001*. Archived from the original (http://cs.nyu.edu/cs/faculty/davism/early.ps) on 2012-07-28. Retrieved 2012-09-08.
8. Bibel, Wolfgang (2007). "Early History and Perspectives of Automated Deduction" (http://www.intellektik.de/resources/OsnabrueckBuchfassung.pdf) (PDF). *Ki 2007*. LNAI (4667). Springer: 2–18. Archived (https://ghostarchive.org/archive/20221009/http://www.intellektik.de/resources/OsnabrueckBuchfassung.pdf) (PDF) from the original on 2022-10-09. Retrieved 2 September 2012.
9. Gilmore, Paul (1960). "A proof procedure for quantification theory: its justification and realisation". *IBM Journal of Research and Development*. **4**: 28–35. doi:10.1147/rd.41.0028 (https://doi.org/10.1147%2Frd.41.0028).
10. McCune, W. W. (1997). "Solution of the Robbins Problem". *Journal of Automated Reasoning*. **19** (3): 263–276. doi:10.1023/A:1005843212881 (https://doi.org/10.1023%2FA%3A1005843212881). S2CID 30847540 (https://api.semanticscholar.org/CorpusID:30847540).
11. Kolata, Gina (December 10, 1996). "Computer Math Proof Shows Reasoning Power" (https://www.nytimes.com/library/cyber/week/1210math.html). *The New York Times*. Retrieved 2008-10-11.
12. Goel, Shilpi; Ray, Sandip (2022), Chattopadhyay, Anupam (ed.), "Microprocessor Assurance and the Role of Theorem Proving" (https://link.springer.com/10.1007/978-981-15-6401-7_38-1), *Handbook of Computer Architecture*, Singapore: Springer Nature Singapore, pp. 1–43, doi:10.1007/978-981-15-6401-7_38-1 (https://doi.org/10.1007%2F978-981-15-6401-7_38-1), ISBN 978-981-15-6401-7, retrieved 2024-02-10
13. Basin, D.; Deville, Y.; Flener, P.; Hamfelt, A.; Fischer Nilsson, J. (2004). "Synthesis of programs in computational logic". In M. Bruynooghe and K.-K. Lau (ed.). *Program Development in Computational Logic*. LNCS. Vol. 3049. Springer. pp. 30–65. CiteSeerX 10.1.1.62.4976 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.4976).
14. Meng, Jia; Paulson, Lawrence C. (2008-01-01). "Translating Higher-Order Clauses to First-Order Clauses" (https://doi.org/10.1007/s10817-007-9085-y). *Journal of Automated Reasoning*. **40** (1): 35–60. doi:10.1007/s10817-007-9085-y (https://doi.org/10.1007%2Fs10817-007-9085-y). ISSN 1573-0670 (https://search.worldcat.org/issn/1573-0670). S2CID 7716709 (https://api.semanticscholar.org/CorpusID:7716709).
15. Bos, Johan. "Wide-coverage semantic analysis with boxer." (https://aclanthology.org/W08-2222.pdf) Semantics in text processing. step 2008 conference proceedings. 2008.
16. Muskens, Reinhard. "Combining Montague semantics and discourse representation." (https://philarchive.org/archive/MUSCMS) Linguistics and philosophy (1996): 143-186.
17. Luckham, David C.; Suzuki, Norihisa (Mar 1976). Automatic Program Verification V: Verification-Oriented Proof Rules for Arrays, Records, and Pointers (https://apps.dtic.mil/sti/citations/ADA027455) (Technical Report AD-A027 455). Defense Technical Information Center. Archived (https://web.archive.org/web/20210812180903/https://apps.dtic.mil/sti/citations/ADA027455) from the original on August 12, 2021.
18. Luckham, David C.; Suzuki, Norihisa (Oct 1979). "Verification of Array, Record, and Pointer Operations in Pascal" (https://doi.org/10.1145%2F357073.357078). *ACM Transactions on Programming Languages and Systems*. **1** (2): 226–244. doi:10.1145/357073.357078 (https://doi.org/10.1145%2F357073.357078). S2CID 10088183 (https://api.semanticscholar.org/CorpusID:10088183).
19. Luckham, D.; German, S.; von Henke, F.; Karp, R.; Milne, P.; Oppen, D.; Polak, W.; Scherlis, W. (1979). *Stanford Pascal verifier user manual* (https://exhibits.stanford.edu/stanford-pubs/catalog/nh154bt5645) (Technical report). Stanford University. CS-TR-79-731.

20. Loveland, D. W. (1986). "Automated theorem proving: Mapping logic into AI". *Proceedings of the ACM SIGART international symposium on Methodologies for intelligent systems*. Knoxville, Tennessee, United States: ACM Press. p. 224. doi:10.1145/12808.12833 (https://doi.org/10.1145%2F12808.12833). ISBN 978-0-89791-206-8. S2CID 14361631 (https://api.semanticscholar.org/CorpusID:14361631).
21. Kerber, Manfred. "How to prove higher order theorems in first order logic (https://kluedo.ub.uni-kl.de/files/364/seki_4.pdf)." (1999).
22. Benzmüller, Christoph, et al. "LEO-II-a cooperative automatic theorem prover for classical higher-order logic (system description) (https://page.mi.fu-berlin.de/cbenzmueller/papers/C26.pdf)." International Joint Conference on Automated Reasoning. Berlin, Germany and Heidelberg: Springer, 2008.
23. Blanchette, Jasmin Christian; Böhme, Sascha; Paulson, Lawrence C. (2013-06-01). "Extending Sledgehammer with SMT Solvers" (https://doi.org/10.1007/s10817-013-9278-5). *Journal of Automated Reasoning*. **51** (1): 109–128. doi:10.1007/s10817-013-9278-5 (https://doi.org/10.1007%2Fs10817-013-9278-5). ISSN 1573-0670 (https://search.worldcat.org/issn/1573-0670). S2CID 5389933 (https://api.semanticscholar.org/CorpusID:5389933). "ATPs and SMT solvers have complementary strengths. The former handle quantifiers more elegantly, whereas the latter excel on large, mostly ground problems."
24. Weber, Tjark; Conchon, Sylvain; Déharbe, David; Heizmann, Matthias; Niemetz, Aina; Reger, Giles (2019-01-01). "The SMT Competition 2015–2018" (https://doi.org/10.3233%2FSAT190123). *Journal on Satisfiability, Boolean Modeling and Computation*. **11** (1): 221–259. doi:10.3233/SAT190123 (https://doi.org/10.3233%2FSAT190123). "In recent years, we have seen a blurring of lines between SMT-COMP and CASC with SMT solvers competing in CASC and ATPs competing in SMT-COMP."
25. Sutcliffe, Geoff. "The TPTP Problem Library for Automated Theorem Proving" (http://www.tptp.org/). Retrieved 15 July 2019.
26. "History" (https://vprover.github.io/history.html). *vprover.github.io*.
27. "The Theorem Prover Museum" (https://theoremprover-museum.github.io). Michael Kohlhase. Retrieved 2022-11-20.
28. Bundy, Alan (1999). *The automation of proof by mathematical induction* (https://www.era.lib.ed.ac.uk/bitstream/handle/1842/3394/0002.pdf?sequence=1) (PDF) (Technical report). Informatics Research Report. Vol. 2. Division of Informatics, University of Edinburgh. hdl:1842/3394 (https://hdl.handle.net/1842%2F3394).
29. Gabbay, Dov M., and Hans Jürgen Ohlbach. "Quantifier elimination in second-order predicate logic." (https://pure.mpg.de/rest/items/item_1834831/component/file_2172339/content) (1992).

References

- Chang, Chin-Liang; Lee, Richard Char-Tung (2014) [1973]. *Symbolic Logic and Mechanical Theorem Proving* (https://books.google.com/books?id=oGriBQAAQBAJ&pg=PR7). Elsevier. ISBN 9780080917283.
- Loveland, Donald W. (2016) [1978]. *Automated Theorem Proving: A Logical Basis* (https://books.google.com/books?id=lsvSBQAAQBAJ&pg=PP11). Fundamental Studies in Computer Science. Vol. 6. Elsevier. ISBN 9781483296777.
- Luckham, David (1990). *Programming with Specifications: An Introduction to Anna, A Language for Specifying Ada Programs*. Springer. ISBN 978-1461396871.
- Gallier, Jean H. (2015) [1986]. *Logic for Computer Science: Foundations of Automatic Theorem Proving* (http://www.cis.upenn.edu/~jean/gbooks/logic.html) (2nd ed.). Dover. ISBN 978-0-486-78082-5. "This material may be reproduced for any educational purpose, ..."

- Duffy, David A. (1991). *Principles of Automated Theorem Proving*. Wiley. ISBN 9780471927846.
- Wos, Larry; Overbeek, Ross; Lusk, Ewing; Boyle, Jim (1992). *Automated Reasoning: Introduction and Applications* (2nd ed.). McGraw–Hill. ISBN 9780079112514.
- Robinson, Alan; Voronkov, Andrei, eds. (2001). *Handbook of Automated Reasoning*. Vol. I. Elsevier, MIT Press. ISBN 9780080532790. II ISBN 9780262182232.
- Fitting, Melvin (2012) [1996]. *First-Order Logic and Automated Theorem Proving* (https://books.google.com/books?id=133kBwAAQBAJ) (2nd ed.). Springer. ISBN 9781461223603.

External links

- A list of theorem proving tools (https://github.com/johnyf/tool_lists/blob/master/verification_synthesis.md#theorem-provers)