Quest Journals
Journal of Research in Applied Mathematics
Volume 6 ~ Issue 5 (2020) pp: 05-10
ISSN(Online) : 2394-0743 ISSN (Print): 2394-0735
www.questjournals.org

Research Paper

The Automation of Mathematical Reasoning

Jollanda Shara
(Department of Mathematics&Computer Science, University "Eqrem Cabej" Albania)

ABSTRACT: *One of the main goals of automated reasoning has been the automation of mathematics. Reasoning is the ability to make inferences, and automated reasoning is concerned with the building of computing systems that automate this process. Although the overall goal is to mechanize different forms of reasoning, the term has largely been identified with valid deductive reasoning as practiced in mathematics and formal logic. Building an automated reasoning program means providing an algorithmic description to a formal calculus so that it can be implemented on a computer to prove theorems of the calculus in an efficient manner. [7] Automated reasoning has made a lot of striking successes over the last decades. It evolved into a rich scientific discipline, with many subdisciplines and with solid grounds in mathematics and computer science. Over the years, automated reasoning transformed from a research field based on mathematical logic into a field that is a driving force for mathematical logic. Nowadays, automated reasoning tools are used in everyday practice in mathematics, computer science and engineering. So, automated reasoning programs are being used by researchers to attack open questions in mathematics and logic, provide important applications in computing science, solve problems in engineering, and find novel approaches to questions in exact philosophy. Also, its role in education increases and will increase in the future.*

KEYWORDS: *Automation, Reasoning, Algebra, Geometry, Automated Theorem Provers*

Received 28November, 2020; Accepted 14 December, 2020
Published with open access at www.questjournals.org

I. INTRODUCTION

The heart of mathematics is setting up abstract problems and solving them, and the outcome of this process is a "proof".

... if we start with an educational purpose,... it is impossible to achieve ideal results by tacking on some additional "interface" features to a previously existing computational system. To put it another way: it is not possible to entirely separate "interface" considerations from "kernel" considerations. (Beeson, 1998)

The formalizability in principle of mathematical proof is the only true controller of correctness. Bourbaki states that "…the correctness of a mathematical text is verified by comparing it, more or less explicitly, with the rules of a formalized language…" (see [3]). Mac Lane [2], too, has pointed out:

"…As to precision, we have now stated an absolute standard of rigor: A mathematical proof is rigorous when it is (or could be) written out in the first-order predicate language $L(\in)$ as a sequence of inferences from the axioms ZFC, each inference made according to one of the stated rules... When a proof is in doubt, its repair is usually just a partial approximation to the fully formal version…" (p377)

However, the idea of formalizing proofs, had seemed impossible before the advent of computers. The main goals of automated reasoning are understanding different aspects of reasoning and development of algorithms and computer programs that solve problems requiring reasoning. Automated reasoning typically combines results and techniques of mathematical logic, theoretical computer science, algorithmics and artificial intelligence. Some subareas of automated reasoning are automated theorem proving, interactive (or formal) theorem proving, automated proof checking, etc. Automated reasoning has applications in software and hardware verification, circuit design, logic programming, program synthesis, deductive databases, ontology reasoning, mathematical software, educational software, robotics, planning, etc. The modern history of automated reasoning is several decades old, but its roots are found many centuries ago when Aristotle and ancient Greek philosophers tried to define forms of reasoning by syllogisms, for reaching at the formal deductive reasoning. Leibniz worked on reducing human reasoning to calculations within symbolic logic that could resolve differences of opinions. His dream was to have a *charateristica universalis* and *calculus ratiocinator* that would allow us to reason in metaphysics and morals in much the same way as this is done in

mathematics. As he stated: "To take pen in hand, sit down at the abacus and, having called in a friend if they want, say to each other: Let us calculate!"

The modern history of automated reasoning starts in early 1950's, along with first programmable computers, and with motivations and key challenges explained by Martin Davis:

...deductive reasoning, especially as embodied in mathematics, presented an ideal target for those interested in experimenting with computer programs that purported to implement the "higher" human faculties. This was because mathematical reasoning combines objectivity with creativity

in a way difficult to find in other domains. For this endeavor, two paths presented themselves. One was to try to understand what people do when they create proofs and to write programs which have to vie that process. The other was to make use of the systematic work of the logicians in reducing logical reasoning to standard canonical forms on which algorithms could be based. There were many difficulties for this. For instance, the well-known unsolvability results of Church and Turing showed that the kind of algorithm on which a programmer might want to base a theorem-proving program simply did not exist. The attractiveness of automated reasoning was also explained by Larry Wos:

"…The beauty of a theorem from mathematics, the preciseness of an inference rule in logic, the intrigue of a puzzle, and the challenge of a game-all are present in the field of automated reasoning."

The key modern international forum for automated reasoning is Association for Automated Reasoning (AAR). The major conferences are Conference on Automated Deduction (CADE), and International Joint Conference on Automated Reasoning (IJCAR), and the major journal is Journal of Automated Reasoning (JAR). Outstanding contributions in the field are honored by Herbrand Award for Distinguished Contributions to Automated Reasoning. Several Turing Awards went to researchers working in the area of automated reasoning or applications of automated reasoning (such as verification): John McCarthy (1971), Allen Newell and Herbert Simon6 (1975), Antony Hoare (1980), Robin Milner (1991), Amir Pnueli (1996), Edmund Clarke, Allen Emerson and Joseph Sifakis (2007).

II. AUTOMATED THEOREM PROVERS

Mathematics is considered as the most abstract of disciplines. Its objects don't respect the boundaries of the physical world. The astronomer Galileo Galilei in his Il Saggiatore wrote that "[The universe] is written in the language of mathematics, and its characters are triangles, circles, and other geometric figures." Thomas Kuhn assigned a special status to mathematics, excluding it from his discrete model of revolution and rupture in the history of scientific knowledge.

By the late twentieth century proof came in new and different forms. Historian of science and technology Donald MacKenzie has argued that automated theorem-proving aroused a debate about what proofs should be like. In particular, he argues that new cultures of proof emerged along two axes. The first axis concerned the kind of proof being produced, which could be either "formal" or "rigorous," the latter also known as "informal." MacKenzie, drawing from the earlier work of Eric Livingston, differentiates between these two kinds of proof as follows:

…A formal proof is a finite sequence of 'well-formed' (that is, to put it loosely, syntactically correct) formulae leading to the theorem, in which each formula is an axiom of the formal system being used or is derived from previous formulae by application of the system's rules of logical inference.... Rigorous arguments, in contrast, are those arguments that are accepted by mathematicians (or other relevant specialists) as constituting mathematical proofs, but that are not formal proofs in the above sense…[4]

All practitioners of automated theorem-proving believed that computers could surprise us. Some early skepticism about the possibilities of computing were grounded on the observation that computers can only do just exactly what they are instructed to do, and that they would therefore never perform beyond what their programmers could imagine for them. However, most early computing practitioners insisted that it was often not possible to know what the consequences of their instructions would be. Indeed, if they could, they would have had little need for fast computing machinery to carry them out. Herbert Simon wrote in 1960 that "…This statement - that computers can only do what they are programmed to do - is intuitively obvious, indubitably true, and supports none of the implications that are commonly drawn from it."

In response to those erroneous implications, two early Artificial Intelligence practitioners Edward Feigenbaum and Julian Feldman summarized the position of most computing practitioners like this:

…It is wrong to conclude that a computer can exhibit behavior no more intelligent than its human programmer and that this astute gentleman can accurately predict the behavior of his program. These conclusions ignore the enormous complexity of information processing possible in problemsolving and learning machines. They presume that, because the programmer can write down (as programs) general prescriptions for adaptive behavior in such mechanisms, he can comprehend the remote consequences of these mechanisms after the execution of millions of information processing operations…(see [4])

The idea of a proof assistant began to draw particular attention in the late 1960s. Many proof assistants were based on a batch model, the machine checking in one operation the correctness of a proof sketch supplied by a human. But a group at the Applied Logic Corporation who developed a sequence of theorem provers in the SAM (Semi-Automated Mathematics) family made their provers interactive, so that the mathematician could work on formalizing a proof with machine assistance. In [8] they write:

…Semi-automated mathematics is an approach to theorem-proving which seeks to combine automatic logic routines with ordinary proof procedures in such a manner that the resulting procedure is both efficient and subject to human intervention in the form of control and guidance. Because it makes the mathematician an essential factor in the quest to establish theorems, this approach is a departure from the usual theorem-proving attempts in which the computer unaided seeks to establish proofs…

Not long after the SAM project, the AUTOMATH [9], Mizar and LCF [10] proof checkers appeared, and many of the most successful interactive theorem provers around today are directly descended from one of these.

So, Automath (de Bruijn 1968) which was the first computer system used to check the correctness of proofs. Automath has been superseded by more modern and capable systems, most notably Mizar. The Mizar system (Trybulec 1979, Muzalewski 1993) is based on Tarski-Grothendieck set theory. Mizar proofs are formal but quite readable, can refer to definitions and previously proved theorems and, once formally checked, can be added to the growing Mizar Mathematical Library (MML) (Bancerek & Rudnicki 2003, Bancerek *et al*. 2018). As of June 2018, MML contained about 12,000 definitions and 59,000 theorems. The Mizar language is a subset of standard English as used in mathematical texts and is highly structured to ensure the production of rigorous and semantically unambiguous texts.

Examples of proofs that have been checked by Mizar include the Hahn-Banach theorem, the Brower fixed-point theorem, Konig's lemma, the Jordan curve theorem, and Gödel's completeness theorem.

Nqthm (Boyer & Moore 1979) has been one of the most successful implementations of automated inductive theorem proving. In the spirit of Gentzen, Boyer and Moore were interested in how people prove theorems by induction. The Boyer-Moore Theorem Prover, [6], represents its control knowledge as procedures for manipulating the mathematical formulae. This theorem prover exploits the relationship between recursion and mathematical induction to guide inductive proofs of the properties of Lisp functions. (see [5]) The recursive definitions of the Lisp functions are first used to symbolically evaluate the conjecture to be proved. For instance, to express the commutativity of addition the user would enter the Lisp expression $(EQUAL\ (PLUS\ X\ Y)\ (PLUS\ Y\ X))$. The Boyer-Moore theorem prover has been used to check the proofs of some quite deep theorems (Boyer, Kaufmann & Moore 1995). Lemma caching, problem statement generalization, and proof planning are techniques particularly useful in inductive theorem proving (Bundy, Harmelen & Hesketh 1991).(see [7])

Thanks to the development of effective algorithms, automated theorem provers have become quite powerful and have achieved remarkable successes. Perhaps the most famous case is McCune's solution [12], using the automated theorem prover EQP, which has been used in 'Robbins conjecture' concerning the axiomatization of Boolean algebra and which had resisted human mathematicians for some time. This success is just one particularly well-known case where the Argonne team has used Otter and other automated reasoning programs to answer open questions.

Automated provers for first-order logic compete against each other in annual competitions on collections of test problems such as TPTP [102], and the Vampire system has usually come out on top for the last few years. There is also intense interest in special provers for other branches of logic, e.g. 'SAT' (satisfiability of purely propositional formulas), which has an amazing range of practical applications. More recently a generalization known as 'SMT' (satisfiability modulo theories), which uses techniques for combining deduction in certain theories has attracted considerable interest. For a nice survey of some of the major interactive systems, showing a proof of the irrationality of $\sqrt{2}$ in each as an example, is given in [11]. (see [1])

Computerized theorem provers can be broken down into two categories. Automated theorem provers, or ATPs, typically use brute-force methods to crunch through big calculations. Interactive theorem provers, or ITPs, act as proof assistants that can verify the accuracy of an argument and check existing proofs for errors. Nowadays there is active and vital research activity in both 'automated' and 'interactive' provers. But these two strategies, even when combined (as is the case with newer theorem provers), don't add up to automated reasoning. In math, theorem provers have helped produce complicated, calculation-heavy proofs that otherwise would have occupied hundreds of years of mathematicians' lives. The Kepler conjecture, which describes the best way to stack spheres (or, historically, oranges or cannonballs), offers a significant example. In 1998, Thomas Hales, together with his student Sam Ferguson, completed a proof using a variety of computerized math techniques. The result was so unhandy - the results took up 3 gigabytes - that 12 mathematicians analyzed it for years before

announcing they were 99% certain it was correct. The Kepler conjecture isn't the only famous question to be solved by machines. The four-color theorem, which says you only need four hues to color any two-dimensional map so that no two adjoining regions share a color, was settled in 1977 by mathematicians using a computer program that churned through five-colored maps to show they could all be reduced to four. And in 2016, a trio of mathematicians used a computer program to prove a longstanding open challenge called the Boolean Pythagorean triples problem, but the initial version of the proof was 200 terabytes in size. With a high-speed internet connection, a person could download it in a little over three weeks. (see [14])

Different proof assistants offer different capabilities measured by their power at automating reasoning tasks, supported logic, object typing, size of mathematical library, and readability of input and output. A "canonical" proof which is not too trivial but not too complex as well can be used as a baseline for system comparison, as done in (Wiedijk 2006) where the authors of seventeen reasoning systems are tasked with establishing the irrationality of $\sqrt{2}$. The systems discussed are certainly more capable than this and some have been used to assist in the formalization of far more advanced proofs such as Erdös-Selberg's proof of the Prime Number Theorem (about 30,000 lines in Isabelle), the formalization of the Four Color Theorem (60,000 lines in Coq), and the Jordan Curve Theorem (75,000 lines in HOL Light). A milestone in interactive theorem proving was reached in 2012 when, after six-years of effort and using the Coq proof assistant, George Gonthier and his team completed the formal verification of the 255-page proof of the Feit-Thompson theorem, also known as the Odd Order Theorem, a major step in the classification of finite simple groups. [7]

III. AUTOMATED REASONING AND ALGEBRA

A computer algebra system (CAS) is a computer program that assists the user with the symbolic manipulation and numeric evaluation of mathematical expressions. Essentially, the computer algebra system operates by taking the input expression entered by the user and successively applies to it a series of transformation rules until the result no longer changes. These transformation rules encode a significant amount of mathematical knowledge making symbolic systems powerful tools in the hands of applied mathematicians, scientists, and engineers trying to attack problems in a wide variety of fields ranging from calculus and the solving of equations to combinatorics and number theory.

Problem solving in mathematics involves the interplay of deduction and calculation, with decision procedures being a reminder of the fuzzy division between the two; hence, the integration of deductive and symbolic systems, which is known as Deductive Computer Algebra (DCA), is bound to be a productive combination. Analytica (Bauer, Clarke & Zhao 1998) is a theorem prover built on top of Mathematica, a powerful and popular computer algebra system. Besides supplying the deductive engine, Analytica also extends Mathematica's capabilities by defining a number of rewrite rules-more precisely, identities about summations and inequalities-that are missing in the system, as well as providing an implementation of Gosper's algorithm for finding closed forms of indefinite hypergeometric summations. Equipped with this extended knowledge, Analytica can prove semi-automatically some nontrivial theorems from real analysis, including a series of lemmas directly leading to a proof of the Bernstein Approximation Theorem.

Rudnicki (2004) discusses the challenges of formalizing Witt's proof of the Wedderburn theorem: Every finite division ring is commutative. The theorem was formulated easily using the existing formalizations available in MML but the proof demanded further entries into the library to formalize notions and facts from algebra, complex numbers, integers, roots of unity, cyclotomic polynomials, and polynomials in general. It took several months of effort to supply the missing material to the MML library but, once in place, the proof was formalized and checked correct in a matter of days. Clearly, a repository of formalized mathematical facts and definitions is a prerequisite for more advanced applications. The QED Manifesto (Boyer *at al.* 1994, Wiedijk 2007) has such an aim in mind and there is much work to do: Mizar has the largest such repository but even after 30 years of work "…it is miniscule with respect to the body of established mathematics" (Rudnicki 2004). This last remark should be construed as a call to increase the effort toward this important aspect in the automation of mathematics.

IV. AUTOMATED REASONING AND GEOMETRY

Automated proving of geometry theorems and solving geometry problems were challenging and inspiring tasks from the beginning of modern computer science. For many hundreds years ago, these sorts of tasks have been considered typical tasks requiring high intellectual skills. One of the very first automated theorem provers was a theorem prover for geometry-Geometry Machine-developed by Herbert Gelertner. This system did not aim at completeness for its domain, but still was able to prove tens of geometry theorems. The system produced traditional, readable Euclidean proofs. It also introduced two innovations (later used in other domains as well): use of symmetries to shorten the proofs and semantic information (obtained from "diagrams") to guide the search. [13]

Significant breakthrough in geometry reasoning came in 1977, when Wen-Tsün Wu introduced an algebraic method (now called Wu's method) capable of proving many complex theorems in Euclidean geometry. With this method, proving of geometry theorems is reduced to solving multivariate polynomial equations. The solving method is based on characteristic sets (introduced by Ritt) and is a decision procedure for certain classes of problems. This theorem prover is sometimes considered the most successful theorem prover overall. Chinese experts selected this method as one of "the four new great Chinese inventions".

Chou (1987) proves over 500 geometry theorems using the algebraic approach offered by Wu's method and the Gröbner basis method by representing hypotheses and conclusions as polynomial equations. Quaife (1992) provides another early effort to find proofs in first-order mathematics: over 400 theorems in Neumann-Bernays-Gödel set theory, over 1,000 theorems in arithmetic, a number of theorems in Euclidian geometry, and Gödel's incompleteness theorems. The approach is best described as semi-automatic or "interactive" with the user providing a significant amount of input to guide the theorem-proving effort. This is no surprise since, as one applies automated reasoning systems into richer areas of mathematics, the systems take more on the role of proof assistants than theorem provers. This is because in richer mathematical domains the systems need to reason about theories and higher-order objects which in general takes them deeper into the undecidable. [7]

In 1965, Bruno Buchberger created the theory of Gröbner bases (named in honor of his PhD advisor, Wolfgang Gröbner), one of the major theories in computer algebra. A Gröbner basis is a particular kind of generating subset of an ideal in a multivariate polynomial ring. Buchberger also designed an algorithm (now known as Buchberger's algorithm) to find a Gröbner basis of a given set of polynomials, i.e., an algorithm for transforming a given set of generators for a polynomial ideal into a Gröbner basis with respect to some monomial order. Gröbner bases can be used for solving simultaneous polynomial equations, for deciding equality of ideals, for deciding membership of ideals, etc. The methodology has many applications in coding theory, cryptography, integer programming and many other areas of mathematics and computer science, including automated theorem proving in geometry. So, hypotheses of a geometry statements can be represented, in algebraic, Cartesian terms, as multivariate polynomials generating an ideal. The conjecture of the statement is valid if the corresponding polynomial belongs to the ideal, hence the problem can be solved by Buchberger's algorithm.

Algebraic methods (like Wu's and Buchberger's one) are very efficient and can prove hundreds of complex geometry theorems. However, the disadvantage is that they do not provide human-readable, traditional proofs, but only yes or no answer, accompanied by an algebraic argument. During 1990's there were several (coordinate-free, nonalgebraic) methods developed that were able to produce more or less readable proofs. Some of them are the area method, and the full angle method. Their main disadvantage compared to the algebraic methods is lower efficiency. Developing automated theorem provers that produce readable proofs efficiently as algebraic provers still remains one of the greatest challenges in the field.

The leading conference in the field is ADG. Some of the modern theorem provers for geometry are GEX/JGEX, Geometry Explorer, and Geo-Proof. Geometry theorem provers have been applied in various scientific and industrial fields, like biology, computer vision, computer-aided design, and robot kinematics. In recent years, geometry provers are integrated into several dynamic geometry systems: computer programs that allow creating, visualizing and anipulating geometric constructions, primarily in plane geometry. [13]

V. CONCLUSION

In mathematics there have been three main revolutions:

1. The introduction of proof by the Greeks in the fourth century BC, culminating in Euclid's Elements.
2. The introduction of rigor in mathematics in the nineteenth century. During this time the non-rigorous calculus was made rigorous by Cauchy and others. This time also saw the development of mathematical logic by Frege and the development of set theory by Cantor.
3. The introduction of formal mathematics in the late twentieth and early twentyfirst centuries.

Many mathematicians are, yet, not aware that this third revolution already has happened, and many probably will disagree that this revolution even is needed. The computer will likely change the game in mathematics as it has done in other scientific fields. ... Once rigorous computer aided and verified proofs are nearly as easy (or easier) than hand generated publishable proofs, they will rapidly become the norm. Work will expand on creating new tools to simplify and automate the process. Mathematicians will be able to work with far more complexity than is practical today. Those who are best able to leverage this capability and to integrate human intuition into this framework will be the most successful…[13]

REFERENCES

[1]. John Harrison, A short survey of automated reasoning**.**
[2]. S. Mac Lane, Mathematics: Form and Function. Springer-Verlag, 1986.
[3]. N. Bourbaki, Theory of sets. Elements of mathematics, Addison-Wesley, 1968.

[4]. Stephanie Aleen Dick, After Math: (Re)configuring Minds, Proof, and Computing in the Postwar United States, 2015.
[5]. Alan Bundy , Discovery and Reasoning in Mathematics
[6]. Boyer, R.S. and Moore J.S., A Computational Logic,Academic Press, ACM monograph series, 1979.
[7]. https://stanford.library.sydney.edu.au/archives/spr2009/entries/reasoning-automated/
[8]. J. R. Guard, F. C. Oglesby, J. H. Bennett, and L. G. Settle, Semi-automated mathematics. *Journal of the ACM*, 1969, 16:49–62.
[9]. N. G. de Bruijn, A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus, and Formalism, 1980, pages 589–606. Academic Press.
[10]. M. J. C. Gordon, R. Milner, and C. P. Wadsworth, Edinburgh LCF: A Mechanised Logic of Computation, volume 78 of Lecture Notes in Computer Science. Springer-Verlag, 1979.
[11]. F.Wiedijk, The Seventeen Provers of theWorld, volume 3600 of Lecture Notes in Computer Science, Springer-Verlag, 2006.
[12]. W. McCune, Solution of the Robbins problem,Journal of Automated Reasoning, 1997, 19:263–276.
[13]. Predrag Janičić, Automated Reasoning: Some Successes and New Challenges, 2011.
[14]. https://www.quantamagazine.org/how-close-are-computers-to-automating-mathematical-reasoning-20200827/

Jollanda Shara. "The Automation of Mathematical Reasoning." *Quest Journals Journal of Research in Applied Mathematics* , vol. 06, no. 05, 2020, pp. 05-10.