Automated Reasoning: Some Successes and New Challenges

Predrag Janičić
Faculty of Mathematics
University of Belgrade
Studentski trg 16, 11000 Belgrade, Serbia
janicic@matf.bg.ac.rs

Abstract. *In this paper a brief account of the area of automated reasoning (or, rather, of some of its subareas) is given. Some historical remarks are given along with overview of some of the most significant results and current challenges. The paper has a somewhat personal perspective, reflecting research interests of the author.*

Keywords. Automated reasoning, automated theorem proving, SAT, SMT, interactive theorem proving

1 Introduction

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 go centuries, or even millennia, back. For instance, Aristotle and ancient Greek philosophers tried to define forms of reasoning by syllogisms, aiming at formal deductive reasoning, while Leibniz worked on reducing human reasoning to calculations within symbolic logic that could resolve differences of opinions. This Leibniz's dream motivated much of the modern mathematical logic, but also much of automated reasoning. 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 [14]:

> ...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 emulating 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. The difficulty with the first approach was that available information about how creative mathematicians go about their business was and remains vague anecdotal. On the other hand, 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. Moreover, it was all too obvious that an attempt to generate a proof of something non-trivial by beginning with the axioms of some logical system and systematically applying

> the rules of inference in all possible directions was sure to lead to a gigantic combinatorial explosion.

The attractiveness of automated reasoning was also explained in a memorable quote by Larry Wos [44]:

> 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).[1] The major conferences are *Conference on Automated Deduction* (CADE),[2] and *International Joint Conference on Automated Reasoning* (IJCAR),[3] and the major journal is *Journal of Automated Reasoning* (JAR).[4] Outstanding contributions in the field are honored by *Herbrand Award for Distinguished Contributions to Automated Reasoning*. Several Turing Awards[5] 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 Simon[6] (1975), Antony Hoare (1980), Robin Milner (1991), Amir Pnueli (1996), Edmund Clarke, Allen Emerson and Joseph Sifakis (2007).

In the rest of this paper, we give a brief account of some areas of automated reasoning. The account is by no means complete, and many results and subareas of automated reasoning (e.g., equational reasoning, term rewriting, inductive reasoning, model checking, proof checking, reasoning in modal, temporal, many-valued, intuitionistic logics, etc) are not discussed. Nevertheless, the given account could hopefully give a hint of richness of this research field. More detailed accounts of automated reasoning and its applications, and of history of automated reasoning can be found in literature [39, 8, 37, 21].

[1] `http://www.aarinc.org/`

[2] `http://www.cadeinc.org/`

[3] `http://www.ijcar.org/`

[4] `http://www.springerlink.com/content/100280/`

[5] The Turing Award, recognized as an equivalent of Nobel prize, is an annual award given by the Association for Computing Machinery (ACM) to "an individual selected for contributions of a technical nature made to the computing community community. The contributions should be of lasting and major technical importance to the computer field."

[6] In 1978. Herbert Simon also received a Nobel prize in economics.

2 Automated Reasoning in Propositional Logic

The propositional logic is the simplest part of mathematical logic, dealing with propositional variables, propositional constants ($\bot$, $\top$) and connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$). Propositional logic has been studied (in some form) even in the ancient Greece, while the major developments came with the work of Augustus DeMorgan and George Boole in the mid-19th century. The first computer program — *Logic Theorist* — able to prove theorem of propositional logic was created in 1956. by Allen Newell, Herbert Simon and Clifford Shaw [35]. The program was one of the very first programs to perform a task for which humans are credited with intelligence. It manipulated not numbers but information represented in symbolic form and the search performed by the program was guided by heuristics. The authors applied the system to Section A of Part I of Whitehead and Russell's monumental *Principia Mathematica* [42] and used the same five axioms as Whitehead and Russell did. The system proved 38 of 52 theorems and for some of them found more elegant proofs (e.g., for the theorem $((p \vee q) \Rightarrow (p \vee r)) \Rightarrow (p \vee (q \Rightarrow r))$). It is interesting that, at the time, one major journal refused to accept a paper describing *Logic Theorist* and its proofs, allegedly because it "would not publish the description of a proof coauthored by a computer program".[7]

Several years after *Logic Theorist* (that used to prove propositional theorems deductively, by using axioms and inference rules), Martin Davis and Hilary Putnam [16] developed a first semantic-based procedure — a decision procedure for checking if the given propositional formula in conjunctive normal form (CNF) was satisfiable[8] (this problem is now called the *SAT problem*; it is obvious that SAT is decidable). This procedure was later improved by

[7] According to some sources, the paper was rejected since a new proof of an elementary mathematical theorem was not notable, apparently overlooking the fact that one of the authors was a computer program.

[8] This procedure was developed in a wider attempt at constructing a proving method for first-order logic.

Martin Davis, George Longmann and Donald Loveland [15] and is usually referred to as the *DPLL procedure*. The procedure, essentially, uses several simple observations to cut the search space and to avoid considering all possible valuations of the involved variables (for a formula involving n propositional variables, there are 2^n valuations to be considered).

Only years after these systems, the notion of NP-complete problems was developed and it was proved by Stephen Cook that SAT is NP-complete problem [13]. Consequently, many hard problems can be solved by reduction to SAT. This motivated further developments in SAT solving and, over the last years, SAT solving was dramatically improved both on the high, algorithmic level and on the implementation level. The DPLL procedure is still in a core of modern solvers for propositional logic, but modern, conflict-driven clause-learning SAT solvers use additional techniques, such as advanced non-chronological search (backjumping) and lemma learning techniques tightly integrated with additional features into coherent systems [5]. Modern SAT solvers typically have a number of parameters that control the used heuristics and that can be tuned to certain classes of input instances (e.g., instances coming from software verification, from computer aided design, from combinatorial problems, etc). Solvers also often differ in some minor aspects, sufficient to lead in significant differences in performance for certain inputs. This led to appearance of *SAT portfolio* solvers that use a number of SAT solvers (or a SAT solver with a number of combinations of parameters) and smart, machine-learning techniques for choosing among them [46, 36]. There is a very active and large SAT solving community with several dedicated conferences,[9] journals,[10] and competitions[11]. Some of the most successful SAT solvers are BerkMin, grasp, MiniSAT, picoSAT, SATzilla, zChaff. Today, state-of-the-art SAT solvers can solve instances (coming from industrial applications) with millions of clauses and thousands of variables. Thanks to these advances, there are a number of practical hard problems that are solved by reduction to SAT. Because of this, a modern SAT solver is often considered a "Swiss army knife" for a wide domain of tasks. Still, reducing different problems to SAT is non-trivial. There are several systems that help in this respect. For example, the system *URSA Major* system uses a specific high-level imperative-declarative language and the specifications are transformed into SAT instances (or to SMT instances), solved by a SAT solver (or by a SMT solver), and the solution is passed back in terms of original variables [34]. The system is suitable for solving a wide class of constraint satisfaction problems. For instance, the system can be used for finding a *seed* value that leads to a specific pseudorandom number in certain iteration — a problem very hard to solve in other programming languages.

Some of the current challenges in SAT solving are checking unsatisfiability proofs of huge input instances, development of verified real-world solvers, development of non-DPLL-based solvers, development of non-CNF solvers, etc.

[9] `http://www.lri.fr/SAT2011/`
[10] `http://jsat.ewi.tudelft.nl/`
[11] `http://www.satcompetition.org/`

3 Automated Reasoning in First-Order Logic

First-order logic (FOL) is more expressive than propositional logic — it allows use of predicate and function symbols and quantification over variables. This expressiveness leads to undecidability — there is no a procedure that can decide whether an arbitrary first-order formula is valid. However, FOL is semi-decidable — there are procedures that can confirm (in a finite number of steps) if an arbitrary first-order formula is valid (if the formula is invalid, then these procedures only in some cases can detect that the formula is invalid, and otherwise they loop). The first such semi-decision procedures were derived by Skolem and Herbrand in 1920s and early 1930s, much before the first computers. Herbrand's procedure was based on unguided search and enumeration of ground terms over, what we nowadays call, Herbrand's universe. This approach is useless for practical applications (i.e., proving non-trivial theorems), but it served as an important step towards practical theorem provers for FOL. One of the first, developed by Paul Gilmore, was based on a refined version of the enumeration [20]. Proving based on this idea used the following structure: theorems are proved by refutation (i.e., the nega-

tion of the theorem is to be checked for inconsistency), the formula being checked for inconsistency is transformed into prenex normal form, existential quantifiers are eliminated in favor of Skolem constants, the quantifier-free part of the formula is transformed into conjunctive normal form, leading to the set of first-order clauses to be checked for inconsistency. For this final (but the hardest) part, the real leap came with Alan Robinson's resolution method [38]. This extremely elegant proof system with only one inference rule raised high expectations among researchers, already stating that automated proving of mathematical theorems too hard for humans is imminent. However, despite many improvements and extensions to the resolution method over time, the first non-trivial conjecture not earlier proved by a human was proved by a computer only in 1997. It was the conjecture that all Robbins algebras are Boolean algebras. It was an open conjecture for more than fifty years and was first proved by EQP (a variant of Otter, a resolution-based theorem prover developed at Argonne National Laboratory). The prover worked over eight days before proving the conjecture. This long-awaited success was one of the central news in media worldwide. After 1997, several other open mathematical conjectures have been proved by automated provers or with their help, especially in finding complicated syntactic proofs with tedious technical steps (e.g., in the domain of quasi-groups and algebraic logic).

The research in first-order theorem proving based on the resolution method and its refinements is still a very active research field. The resolution method's legacy includes logic programming and the language Prolog and its variants. Modern resolution based provers can decide problem instances with thousands of variables and clauses. There is a large database of problem instances TPTP for first-order theorem proving [41], used for evaluation of provers and for competitions.[12] Some of the most successful theorem provers for first-order logic are E, Otter/Prover9, Spass, Vampire.

Apart from resolution, a number of other methods (such a the tableaux method) have been also extensively used in automated theorem proving. However, there is a general concern: despite the fact that these *uniform proof procedures* give elegant semi-decision procedures for FOL, they are often helpless with conjectures in specific first-order theories, important for applications, such as linear arithmetic. Moreover, some of such specific first-order theories are decidable so they admit proper decision procedures. Specific procedures for specific first-order theories can be much more efficient than general procedures such as the resolution method. The first automated theorem prover specialized for one first-order theory — linear arithmetic — was developed by Martin Davis in 1954, following the decision procedure due to Presburger. According to Davis, "its great triumph was to prove that the sum of two even numbers is even" [14]. Over the next decades, especially after 2000., a lot of efforts have been invested in developing efficient specialized decision procedures. This field is now called "satisfiability modulo theories" (SMT). SMT solvers are based on SAT solvers in combination with solvers able to check consistency of a set of literals in the underlying theory. Theories most often used are: linear arithmetic, theory of uninterpreted functions, theory of arrays, bit-vector arithmetic, etc. There are schemes for combining decision procedures into a decision procedure for the combined theory, if certain conditions are met [23]. Modern SMT solvers are successfully used in solving a wide range of problems, primarily in software and hardware verification, planning, scheduling, etc. They can decide instances from industrial applications that take gigabytes to get stored. There is a workshop dedicated to SMT solving[13] and an annual competitions for SMT solvers.[14] Some of the most successful ones are Boolector, MathSAT, Yices, Z3.

One of the most important field of application of automated reasoning in first order logic (but also in other logics, such as temporal logics) and especially of SMT solvers are in software and hardware verification. An estimated annual cost of the bugs in software and hardware to the US economy alone is between 20 and 60 billion dollars. Although automated reasoning was used in verification tasks for decades, real successes and real-world applications came only in 1990s. Major software and hardware companies like Intel, Microsoft, NEC, AMD, IBM have been using and developing tools for formal verification since mid-1990's

[12] `http://www.cs.miami.edu/~tptp/CASC/`

[13] `http://uclid.eecs.berkeley.edu/smt11/`

[14] `http://www.smtcomp.org`

[18, 17, 3]. In the meanwhile, many complex software components were successfully verified and verification tools are used in developing safety-critical applications in many companies. However, one of the most important goals for automated reasoning remains development of robust and reliable verification tools that will be regularly used in everyday practice. Some of these goals are formulated within a *Verification Grand Challenge* programme, advocated since 2005. by Tony Hoare's and envisioning a world where programs would only be produced with machine-verified guarantees of adherence to specified behavior. The programme was inspired and comparable to the human genome project (1990-2004). It has a 15 year perspective and would require over a thousand person-years of skilled scientific effort from all over the world.

4 Automated Reasoning in Higher-Order Logic and Interactive Theorem Proving

Higher-order logic is more expressible than first order logic as it admits several types of quantification. For instance, it is not only quantification over individual variables that is allowed, but also quantification over predicate and function symbols. Such expressive framework is suitable for some tasks but efficient automated theorem prover for it is typically a big challenge.

In automated reasoning, higher-order logic is used also as a setting for interactive theorem proving. Interactive theorem proving systems (or *proof assistants*) are used to check (and guide) proofs constructed by the user, by verifying each proof step with respect to the given underlying logic. Proofs constructed within proof assistants are verbatim and detailed, and typically much longer than "traditional proofs".[15] On the other hand, "traditional proofs" most often are not proofs at all, because of the many missing fragments, informal arguments, etc. Using interactive theorem proving uncovered many flaws in many published mathematical proofs (including some seminal ones), published in books and journals. Some of the most popular modern proof assistants are Isabelle, Coq, HOL Light, PVS, Mizar, ACL2, etc [43]. The leading conference in the field is ITP.[16]

When checking proofs, correctness of proof assistants themselves is also critical. The theorem provers based on the LCF tradition (such as Isabelle and HOL Light) have a very small kernel that checks all other derivations (this is called the "de Bruijn criterion"). This small core can be just tens of lines of code and can be manually verified.

Interactive theorem proving gets more and more popular and its expected role in the future is illustrated by the following Wiedijk's comments:

> 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 twenty-first centuries.[17]
>
> Most mathematicians are 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

[15] The ratio of formal proof length to informal proof length is often called the *de Bruijn factor* and it varies for different systems. It is often around 4 [4].

[16] `http://itp2011.cs.ru.nl/ITP2011`

[17] Here Wiedijk presumably thinks of "computer supported formal mathematics". Namely, formal mathematics and formalist theory appeared one century earlier — in the late nineteen and early twentieth centuries.

> 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.

In 1993, an idea for a large-scale international project QED (Q.E.D. stands for "quod erat demonstrandum" in Latin, i.e., "that which was to be demonstrated") aimed at formalization of mathematical knowledge was presented by Robert Boyer and other researchers. The goals of the project were outlined in 1994, in the *QED manifesto* [2], calling for a computer-based database of all important, established mathematical knowledge, strictly formalized and with all proofs having been checked automatically. The project would be a major scientific undertaking requiring the cooperation and effort of hundreds of mathematics and computer scientists, research groups, research agencies, universities, and corporations. This system will have benefits for mathematics, science, technology, and education. Although a formal project has never been started, many researchers and QED-style projects follow the goals presented in the manifesto and many of them have been achieved in the meanwhile. There are several theorems proved for the first time thanks to proof assistants (e.g., Higman's lemma and analysis of Gerard's paradox [1]). From the (informally and somewhat arbitrarily compiled) list of "top 100 mathematical theorems", 86 have been formalized so far.[18] These proofs include proofs of complex classical theorems like prime number theorem (formalized within HOL Light and within Isabelle), Gödel's Incompleteness Theorem (formalized within Coq, Isabelle, Nqthm), Sylow's Theorem (within Coq, Mizar, Isabelle), etc. Among these proofs is also a proof of the "four color theorem" (given any separation of a plane into contiguous regions, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color). The theorem was proved in 1976. by Kenneth Appel and Wolfgang Haken as the first major theorem to be proved using a computer (but without a formal proof verified by some proof assistant). Their proof was largely combinatorial and considered a particular set of 1936 maps. Initially, their proof was not accepted by all mathematicians because the computer-assisted proof was infeasible for a human to check by hand. In 2005, the theorem was proved (within Coq) by Georges Gonthier, dismissing all concerns about the validity of the proof.

Another interesting example of applications of interactive theorem proving concerns the proof of the Kepler Conjecture. The Kepler conjecture says that no packing of congruent balls in Euclidean three space has density greater than that of the face-centered cubic packing — $\pi/\sqrt{18} \approx 0.74048$. This is the oldest problem in discrete geometry and is an important part of Hilbert's 18th problem. It remained unsolved for nearly 400 years until it was finally solved in 1998. by Thomas Hales. Hales' proof, one of the most complicated mathematical proofs ever produced, was based on checking of many individual cases using complex computer calculations. The journal *Annals of Mathematics* solicited the paper for publication in 1998. and hosted a conference in 1999. that was devoted to understanding the proof. A panel of 12 referees was assigned to the task of verifying the correctness of the proof and after four full years, it returned a report stating that they were 99% certain of the correctness of the proof, but they unable to completely certify the proof (still, the journal published the proof in 2005). In order to dismiss concerns about his proof, Hales in 2003 initiated the *Flyspeck* project to fully formalize (within HOL Light) his proof and all computer programs used in constructing it. It is involves a number of researchers and is expected to take 66 man-years.[19]

Formalization of mathematical knowledge is important for dealing with both classical and modern mathematical proofs (especially if they involve complex combinatorial arguments provided by computer) and for education — for deeper understanding of mathematics. It is also very important in software and hardware verification for verifying safety-critical computer programs. Some of the applications of interactive theorem proving in verification are: correctness of the floating point divide operations for AMD's Pentium-like AMD5K86 microprocessor was proved (within ACL2); a microprocessor for aircraft flight control has been verified (within PVS); correctness of the FM9001 microprocessor was proved (within Nqthm); correct-

[18] `http://www.cs.ru.nl/~freek/100/`

[19] `http://code.google.com/p/flyspeck/wiki/FlyspeckFactSheet`

ness of a modern SAT solvers was proved (within Isabelle)[31, 32, 33].

The role of interactive theorem proving is not limited to mathematics and computer science. Actually, many other reasoning tasks can be formalized (in mathematical terms) and treated within proof assistants. One such example is analysis of chess problems: for instance, retrograde chess analysis [30] or analysis of correctness of strategies for chess endgames [29].

Despite all results and successes of interactive theorem proving, there are still very few mathematicians using it. Therefore, one of the main challenges is to further develop theorem provers so they are easier to use and more simply resemble traditional mathematics [9]. There are steps in these directions, necessary to make interactive theorem proving more easily accessible to mathematicians.

5 Automated Reasoning in Geometry

Automated proving of geometry theorems and solving geometry problems were challenging and inspiring tasks from the very first years of modern computer science. Namely, for millennia these sorts of tasks have been considered typical tasks requiring high intellectual skills. There is a (algebraic-based) decision procedure (due to Tarski) for a variant of Euclidean geometry but it is practically useless for non-trivial theorems. One of the very first automated theorem provers was a theorem prover for geometry — *Geometry Machine* — developed by Herbert Gelertner [19]. 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.

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 [45], including many problems from International Mathematical Olympiads [10]. 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 (not only in the field of geometry). Chinese experts selected this method as one of "the four new great Chinese inventions".[20]

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 [6]. 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 [7], including automated theorem proving in geometry (the same class of theorems dealt with by Wu's method) [28]. Namely, 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 drawback 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, non-algebraic) methods developed that were able to produce more or less readable proofs. Some of them are the area method [12, 27], and the full angle method [12]. Their main disadvantage compared

[20] The remaining three "modern great inventions" are hybrid rice, synthesized crystalline bovine insulin, and land facies oil-forming theory. The four ancient "great Chinese inventions" are papermaking, printing, gunpowder and compass (`http://www.edu.cn/20060215/3173112.shtml`).

to the algebraic methods is lower efficiency. There are approaches, combining reasoning in coherent logic with SAT solving, aiming at producing traditional, readable and formal proofs efficiently for some classes of geometry theorems [26, 40]. 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.[21] Some of the modern theorem provers for geometry are GEX/JGEX, Geometry Explorer, and GeoProof.

Geometry theorem provers have been applied in various scientific and industrial fields, like biology, computer vision, computer-aided design, and robot kinematics [11]. In recent years, geometry provers are integrated into several *dynamic geometry systems* — computer programs that allow creating, visualizing and manipulating geometric constructions, primarily in plane geometry. One of the dynamic geometry systems with integrated theorem proving is a system GCLC [22], based on a custom *geometry construction language* [25]. There are three geometry theorem provers (based on the Gröbner bases method, on Wu's method, and on the area method) built in GCLC, all capable of efficiently proving hundreds of complex geometry theorems [24].

6 Conclusions

Automated reasoning has made a lot of striking successes over the last fifty years. 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. Also, its role in education (both as a supporting tool and a subject matter) increases and should increase further.

In this essay we gave just a very short account of the development of automated reasoning and mentioned only a fraction of successes of this fascinating field. More successes of automated reasoning are still to come, and in this essay some of the challenges and goals were also briefly discussed.

[21] `https://lsiit.u-strasbg.fr/adg2010`

Acknowledgements. The work on this paper was partially supported by the Serbian Ministry of Science grant 174021. I am grateful to Filip Marić, Mladen Nikolić and Mirko Čubrilo for useful comments on an earlier version of this paper.

References

[1] Stuart F. Allen, Mark Bickford, Robert L. Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and E. Moran. Innovations in computational type theory using Nuprl. *J. Applied Logic*, 4(4), 2006.

[2] Anonymous. The QED manifesto. In *Proceedings of the 12th International Conference on Automated Deduction — CADE-12*, volume 814 of *Lecture Notes in Computer Science*, pages 238–251. Springer, 1994.

[3] Gogul Balakrishnan, Malay K. Ganai, Aarti Gupta, Franjo Ivancic, Vineet Kahlon, Weihong Li, Naoto Maeda, Nadia Papakonstantinou, Sriram Sankaranarayanan, Nishant Sinha, and Chao Wang. Scalable and precise program analysis at NEC. In Roderick Bloem and Natasha Sharygina, editors, *Proceedings of 10th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2010*, pages 273–274. IEEE, 2010.

[4] Henk Barendregt and Freek Wiedijk. The challenge of computer mathematics. *Philosophical Transactions of the Royal Society*, 363(1835):2351–2375, 2005.

[5] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[6] Bruno Buchberger. *An Algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*. PhD thesis, University of Innsbruck, Austria, 1965.

[7] Bruno Buchberger and Franz Winkler, editors. *Gröbner Bases and Applications*. Cambridge University Press, 1998.

[8] Alan Bundy. A survey of automated deduction. In Wooldridge M. J. and Veloso M., editors, *Artificial Intelligence Today*, pages 153–174. Springer, 1999.

[9] Alan Bundy. Automated theorem provers: a practical tool for the working mathematician? *Annals of Mathematics and Artificial Intelligence*, 61(1):3–14, 2011.

[10] Shang-Ching Chou. *Mechanical Geometry Theorem Proving*. D.Reidel Publishing Company, Dordrecht, 1988.

[11] Shang-Ching Chou and Xiao-Shan Gao. Automated reasoning in geometry. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[12] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. *Machine Proofs in Geometry*. World Scientific, Singapore, 1994.

[13] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM Press, 1971.

[14] Martin Davis. The early history of automated deduction. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[15] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[16] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *Journal of Association for Computing Machinery*, 7(3):201–215, 1960.

[17] Leonardo Mendonça de Moura and Nikolaj Bjørner. Bugs, moles and skeletons: Symbolic reasoning for software development. In Jürgen Giesl and Reiner Hähnle, editors, *Proceedings of 5th International Joint Conference on Automated Reasoning – IJCAR 2010*, volume 6173 of *Lecture Notes in Computer Science*, pages 400–411. Springer, 2010.

[18] Limor Fix. Fifteen years of formal property verification in Intel. In Orna Grumberg and Helmut Veith, editors, *25 Years of Model Checking*, volume 5000 of *Lecture Notes in Computer Science*, pages 139–144. Springer, 2008.

[19] Herbert Gelernter. Realization of a geometry theorem proving machine. In *Proceedings of the International Conference Information Processing*, pages 273–282, Paris, June 15-20 1959.

[20] Paul Gilmore. A proof method for quantification theory: its justification and realization. *IBM Journal of Research and Development*, 4:28–35, 1960.

[21] Ben Goertzel, Nil Geisweiller, Lucio Coelho, Predrag Janičić, and Cassio Pennachin. *Real-World Reasoning: The Challenge of Scalable, Uncertain Spatiotemporal, Contextual and Causal Inference*. Atlantis Press, 2011.

[22] Predrag Janičić. GCLC – A Tool for Constructive Euclidean Geometry and More than That. In Nobuki Takayama, Andres Iglesias, and Jaime Gutierrez, editors, *Proceedings of International Congress of Mathematical Software (ICMS 2006)*, volume 4151 of *Lecture Notes in Computer Science*, pages 58–73. Springer-Verlag, 2006.

[23] Predrag Janičić and Alan Bundy. A general setting for flexibly combining and augmenting decision procedures. *Journal of Automated Reasoning*, 28(3):257–305, 2002.

[24] Predrag Janičić and Pedro Quaresma. System description: GCLCprover + GeoThms. In Ulrich Furbach and Natarajan Shankar, editors, *International Joint Conference on Automated Reasoning (IJCAR-2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 145–150. Springer-Verlag, 2006.

[25] Predrag Janičić. Geometry Constructions Language. *Journal of Automated Reasoning*, 44(1-2):3–24, 2010.

[26] Predrag Janičić and Stevan Kordić. EUCLID — the geometry theorem prover. *FILOMAT*, 9(3):723–732, 1995.

[27] Predrag Janičić, Julien Narboux, and Pedro Quaresma. The area method: a recapitulation. *Journal of Automated Reasoning*. To appear.

[28] Deepak Kapur. Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, 1986.

[29] Marko Maliković, Predrag Janičić, and Mirko Čubrilo. Formal analysis of correctness of strategies for chess endgames. In preparation.

[30] Marko Maliković and Mirko Čubrilo. What were the last moves? *International Review on Computers and Software*, 5(1):59–70, 2010.

[31] Filip Marić. Formalization and Implementation of Modern SAT Solvers. *Journal of Automated Reasoning*, 43(1):81–119, 2009.

[32] Filip Marić. Formal verification of a modern SAT solver by shallow embedding into isabelle/hol. *Theoretical Computer Science*, 411(50):4333–4356, 2010.

[33] Filip Marić and Predrag Janičić. Formalization of abstract state transition systems for SAT. *Logical Methods in Computer Science*. To appear.

[34] Filip Marić and Predrag Janičić. Urbiva: Uniform reduction to bit-vector arithmetic. In *Automated Reasoning (IJCAR 2010)*, volume 6173 of *LNCS*, pages 346–352. Springer, 2010.

[35] Allen Newell, J. Clifford Shaw, and Herbert Simon. Empirical explorations with the logic theory machine. In Jörg Siekmann and Graham Wrightson, editors, *Automation of Reasoning: Classical Papers on Computational Logic*, pages 49–73. Springer, 1983. Originally published in *Proceedings of the Western Joint Computer Conference*, 1957.

[36] Mladen Nikolić, Filip Marić, and Predrag Janičić. Instance-based selection of policies for SAT solvers. In *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 326–340. Springer, 2009.

[37] Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[38] J. A. Robinson. A machine oriented logic based on the resolution principle. *Jorunal of ACM*, 12:23–41, 1965.

[39] J. Siekmann and G. Wrightson, editors. *Automation of Reasoning: Classical Papers on Computational Logic*. Springer, 1983.

[40] Sana Stojanović, Vesna Pavlović, and Predrag Janičić. Automatic verification of regular constructions in dynamic geometry systems. In *Automated Deduction in Geometry*, Lecture Notes in Artificial Intelligence, 2011. To appear.

[41] Geoff Sutcliffe. The TPTP world - infrastructure for automated reasoning. In Edmund M. Clarke and Andrei Voronkov, editors, *16th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning – LPAR-16*, volume 6355 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2010.

[42] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. Cambridge University Press, 1910, 1912, and 1913.

[43] Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.

[44] Larry Wos. *Automated Reasoning: 33 Basic Research Problems*. Prentice Hall, 1988.

[45] Wen-Tsün Wu. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, 21:157–179, 1978.

[46] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008.