TLP: page 1 of 6 © Cambridge University Press 2015
doi:10.1017/S1471068415000125

Book review

All about Proofs, Proofs for All, Bruno Woltzenlogel Paleo and David Delahaye, Eds., College Publications, Series Mathematical Logic and Foundations, vol. 55., 2015. Paperback, ISBN 978-1-84890-166-7, vii + 240 pages.

1 Introduction

This book is a collection of articles referring to the presentations delivered during a one-day workshop consisting of 12 invited tutorials. That workshop was a part of "Vienna Summer of Logic" in July 2014. The format itself (I did not attend the event) indicates tutorials of a moderate length devoted to a multitude of deduction tools.

The reviewed volume consists of twelve chapters, i.e. papers written by presenters and reviewed by individuals from the corresponding community and other communities. College Publications, a publisher of all-things-logic[1] produced the book fast; the date on the preface is less than half-a-year after the event. This included the review process! To some extent, this is not surprising. With the appropriate tools and printing-on-demand process the scientific community can now liberate itself from mercy of shark-publishers. College Publications shows the way.

The chapters in the collection correspond to tutorials presented during the workshop. The tools (and more generally subspecialties of computational logic) presented in the chapters range from provers associated with some areas of propositional logic, through the provers for decidable fragments of predicate logic, to first-order logic provers, and higher-order logic. Applications of proofs, more generally of Proof Theory, are also discussed.

There are significant differences between the tools and, more generally, philosophies of corresponding communities. The border lies between so-called satisfiability-modulo-theories, i.e. decidable fragments of predicate logic (I am being deliberately not precise here) and more complex areas of logic: full first-order logic, and higher-order logics.

Generally, since antiquity, the humanity associated values with *proofs*. Certainly the ancient Greeks understood that, and the proofs and proof techniques were followed by philosophers and mathematicians of antiquity. Techniques that did not perish during the dark ages were further developed by mathematicians and philosophers starting with the mediaeval scholastics. Of course other great civilizations, Arabs, Chinese, Indians, and Persians, contributed, in various degrees, in preservation and developments of such techniques. Modern mathematics is founded on the notion of a proof. Proofs are what distinguishes theorems from conjectures.

[1] Full disclosure: this reviewer published with them.

Advent of computers resulted in situations where proofs are claimed, but not necessarily provided. Let us look at something that relates to every computer user, specifically one issue in computer security. Modern computers (for instance the laptop on which you, the reader, are reading this review) have cryptographic co-processors, digital circuits that compute the so-called AES encryption. This functionality is crucial for secure communication with your healthcare provider, or an e-commerce company such as eBay. The designers of circuits write (in a suitable formal language) the description of such co-processor. Then, the producers of the circuit optimize the design and send it to a *foundry* where the circuit will be mass-produced. The optimization changes the design to some extent. Now, how do we know (to be precise: how the vendor of the laptop knows) that the actual implementation is functionally equivalent (i.e. produces the correct output on each input) to the specification? Certainly, the society is vitally interested knowing that the implemented circuit does what the specification describes — nothing more and nothing less. In principle, this requires 2^{128} checks (one for each key). But maybe, just maybe, we could *prove* that the specification and implementation (both converted into a suitably chosen formal language) are equivalent?

The application I mentioned above seems to indicate that at least some (very practically minded) communities may benefit from practical applications of proof theory: specifically, computer engineers. Certainly other specialities (for instance cryptographers) can use tools originating from proof theory; the last chapter of the collection gives an overview of the use of computer-aided proofs in Cryptography.

2 Contents of the collection

Now, let us discuss specific chapters in the collection. The first one, by M. Heule and A. Biere *Proofs for Satisfiability Problems* presents the role of proofs in SAT (satisfiability). Since SAT is perhaps the most important tool for solving finite-domain constraint satisfaction problems the question of getting proofs of results in this area are paramount. To simplify slightly, it is really about getting proofs of inconsistency when the (clausal) theory is unsatisfiable. The paper describes the recent progress in this area. While it is clear that many classical problems in extremal combinatorics can still not be handled by SAT solvers, the progress is very fast and we can expect that within a reasonable period of time many such problems will be solved.

Often SAT is used as a tool for finding solutions using decidable fragments of predicate logic. Again, not finding a solution means that the theory in such fragment is unsatisfiable. The issue, then, is to find the unsatisfiability proof (again this is really a general case to which provability is reduced). This area of solving constraint satisfaction is called Satisfiability modulo Theories (SMT). The second chapter of the volume, "*Proofs in Satisfiability Modulo Theories*", by C. Barrett, L. de Moura and P. Fontaine discusses both the story of SMT, and technology for solving constraint satisfaction problems with SMT as well as proof extraction (when such problems are unsolvable). Specific solvers are discussed.

The chapter "*Proof Generation for Saturating First-Order Theorem Provers*" by S. Schulz and G. Sutcliffe, discusses proofs extracted from first-order theorem provers. The paper provides a limited information about first-order theorem provers (see remarks at the end of this review) and discusses proof formats for the proofs extracted from the operation of provers. The language TPTP (for both problem descriptions and solutions, i.e. proofs) associated with the work of both authors is discussed. Numerous and meaningful examples of problems and proofs are given.

The next chapter, "*Higher-Order Automated Theorem Provers*", by Ch. Benzmüller introduces the reader to proofs in *HOL*, a proof system for higher-order logic. This system allowing for higher-order formulas and quantifications over higher-order objects handles significant extension of first-order logic. It is known that many formalisms can be embedded into *HOL*. A number of specific systems such as *Leo-II*, *Isabelle*, and several other proof systems based on higher-order logic are discussed. Both automated and interactive theorem provers for higher-order logic are presented.

The chapter "*Interactive Theorem Proving from the perspective Isabelle/Isar*" by M. Wentzel returns to the higher-order logic theorem provers, discussing in detail interactive theorem prover *Isabelle* and its "relatives": *Coq* and *HOL*. An interested reader will find a lot of details both about the syntactic conventions used in *Isabelle* and related systems. The efforts to integrate tools of SMT family (for instance *Z3*) are presented. There is an extensive discussion of the proof format, hence Isar, and an actual formalized proof (of Tarski fixpoint theorem: the meet of all prefixpoints of a monotone operator O in a complete lattice is the least fixpoint of O.)

The next chapter in the collection, "*Introduction to the Calculus of Inductive Constructions*" by Ch. Paulin-Mohrig provides the overview of the formalism that is the theoretical basis of *Coq* proof assistant. It is an informative account of the basic properties of inductive constructions, providing wealth of information on syntax and reasoning behind *Coq*.

"*Deduction modulo theory*" by G. Dowek, describes the following idea: when we prespecify theory T and get the unary predicate $\vdash_T$ on the set of formulas (of a fixed signature σ_T), this predicate itself can have nicer computational properties than the provability predicate $\vdash$. One can think about systems based on such idea as a lifting of the SMT systems to the predicate calculus case. The specialized theory T determines the equivalence relation $\equiv_T$ (namely $T \vdash \alpha \equiv \beta$) and this relationship can be used in rewrite rules. The paper gives a short description of the system *Dedukti*, based on this idea.

One of the fundamental issues in the area of Automated Theorem Proving (ATP) is lack of standards for proofs (and other related predicates). The point is that formalization of the notion of proof is, in general, not something mathematicians do. There may be various reasons, but the truth is that mathematicians have been doing proofs for thousands of years and we accept proofs of, say, Euclid, as perfectly correct. In fact, mathematicians have been, generally, unhappy with the formalizations of the proofs, often considering work in this area as *philosophy* (used in derogatory sense). But in ATP standards are needed. The chapter "*Foundational*

Proof Certificates" by D. Miller treats this area and provide an extensive discussion of the efforts aiming at building the *flexible* framework for proof description.

The chapter "*Deep Inference*" by A. Guglielmi deals with the algebraic properties of proofs in the tradition of linear logic.

Next, A. Leitsch in his "*On proof-mining by cut-elimination*" presents the technique of utilizing resolution as means to analyze proofs that use cut elimination. An interesting application — the analysis of Fürstenberg proof of existence of infinite number of primes — is given. The ATP system *ceres* is discussed in the paper.

The paper by J.-R. Abral, "*Definition of a Mathematical Language Together with the Proof System in Event-B*" presents the *Rodin* platform. This platform contains a variety of provers, including various SMT systems and their own predicate logic prover. The platform is used to test correctness of embedded industrial systems. Specific industrial applications are mentioned.

Finally, the chapter "*Computer-aided cryptography: some tools and applications*" by G. Barthe, F. Depressoir, B. Grégoire, B. Schmidt, and P.-Y. Strub, describes a system called *CertiCrypt* that uses *Coq* proof-assistant to provide probabilistic proofs of security properties. This is a novel application of formal methods, but the proofs so derived are not the objects that a proof theorist normally accepts as a *proof*; the way I see it is a technique to assign probabilities to security of cryptographic constructions. Examples given in the paper include analysis of encryption schemes and analyzing properties of schemes based on the difficulty of computing discrete logarithm.

3 Discussion

Now, that we looked at the contents of the book, we need to ask ourselves questions related to the utility of this book. Generally, over the millennia, mathematicians and philosophers used the fact that a formula has a proof, as a criterion of validity. In fact, the existence of a proof was (and still is) a major criterion of acceptance of mathematical formulas as correct. But the availability of computers changed the situation to an extent. Let us now discuss one example of such a spectacular result. M. Kouril and J.L. Paul (The van der Waerden Number $W(2,6)$ is 1132. *Experimental Mathematics* 11:53–61, 2008) found that whenever the integer segment [1..1132] is split into two blocks, at least one of these blocks contains an arithmetic progression of length 6. This is an amazing fact (previously Kouril and Franco found that the number 1131 does not have the same property). Paul Erdős would certainly be thrilled! But is an average mathematician going to believe it? After all, what happened was that a custom SAT solver, implemented using a couple of FPGA (reconfigurable circuits) searched the space of all partitions of [1..1132] into two blocks and did not find a partition without arithmetic progression of length 6 in at least one block. Thus it is claimed that (the representation of) the problem is unsatisfiable. The computation took ca. 9 months of continuous operation. We can safely trust that the representation of the problem is correct. Even if we extract a collection of clauses that are a consequence of the representation of the problem and check (using a computer, as the collection of these clauses is too big to be handled

"by hand") that it is unsatisfiable a (very small) possibility of an error persists! Still we can be reasonably convinced that the representation is, in fact, unsatisfiable and that the desired result holds. Several examples of this kind (including the famous McKay and Radziszowski result on number $R(4,5)$) are known. So, we need proofs: mathematicians, but the rest of the society as well. This was, certainly understood for some 60 years now, and significant progress of ATP ensued. But there is an important difference between the way mathematicians function and the ATP. Quite often, mathematicians omit steps that are (in their opinion) obvious. Trusting their intuitions, many mathematicians made mistakes, including some famous ones (nomina sunt odiosa ...) For that reason proof assistants (such as *Coq* and *MIZAR*) attempt to "fill details". A spectacular and important application of this technology was the work of Gonthier of Microsoft Research (Gonthier 2005) with the claim that using one of the versions of *Coq* he was able to verify the proof of Four-color theorem[2] published by Robertson *et al.* (1997). Completion of yet another important verification effort (this time of Hales proof of Kepler Conjecture) called *flyspec*, has been announced in (Flyspeck 2014). We point, however, that the proofs have been completed/checked using computer programs, and so the (small) doubts of correctness of *checking* (see our discussion in Section 1) must persist.

4 Coda

Now, the question is if the reviewed book reflect that 60 years progress. Specifically, if for whatever reason you, the reader, need to gain an additional certainty of some property, are you going to turn to one of the systems described in this collection? In other words, is this panorama of ATP complete? My answer to this question is that it is not. While we get a significant amount of information on extraction of proofs from operation of SAT solvers and SMT solvers, the situation in case of systems dealing of formulas admitting quantifiers (thus variables) is significantly different from one (implicitly) presented in this collection. While lip-service is paid to systems not presented in this collection (some short references, here-and-there), the industrially-strong systems such as ACL2 and its predecessor NQTHM, out of University of Texas, Austin and PVS out of SRI are not discussed. This is truly amazing, as the first of these was a subject of the ACM Software Award (Boyer, Kaufmann and Moore 2005) and both of these systems have a long history of significant *industrial* applications, significant communities of users and developers. As concerns proof assistants (certainly *Coq* is discussed in the collection in several places) there is no discussion of the *MIZAR* proof assistant, a system with a significant following in a variety of places.

Word-processing by the authors, stylefiles, etc. makes production of such collections relatively easy (this reviewer had his share of such ventures); the editors need to put things together and make sure that word-processing and other formal properties of the document look right. However, a paper that contains 33 citations

[2] Gonthier acknowledges work of B. Werner.

of papers authored or coauthored by the (single) author of the chapter in question — I am not making it up — is a bit of an anomaly and its presence indicates that the editors did not do a good job.

Victor W. Marek
University of Kentucky
(*email: marek@cs.uky.edu*)

References

Flyspeck. 2014. The Flyspeck Project, T. Hales, project director. Accessed 10 July 2015. URL: http://code.google.com/p/flyspeck/wiki/AnnouncingCompletion.

Gonthier, G. 2005. A computer-checked proof of the four-colour theorem. Accessed 10 July 2015. URL: http://research.microsoft.com/en-us/um/people/gonthier/4colproof.pdf.

Robertson, N., Sanders, D., Seymour, P. and Thomas, R. 1997. The four-colour theorem. *Journal Combinatorial Theory Series B* 70, 2–44.