CADE 2021

Notes by J.H.Davenport

11-16 July 2021

Contents

Chapter 1

ThEDU — 11 July

1.1 Gilles Dowek — Why should we teach formal proof to a large audience?

Applications, notably safety. Maths, notably Hales. Also a moral duty in "post-truth era". Note that mathematicians understand that truth is relative: "$\exists x : x^2 = 2$" is false in $|Q$ but true in $\mathbf{R}$; to the sum of the angles of a triangle add to π — only with Euclid's axiom; Lebesgue nonmeasurable sets depends on AC; if f iscontinuous first neagtvie then positive, can we compute the 0 — needs excluded middle.

30 year ago the challenge was MSc in CS, now it's to undergraduates in both CS and Maths. Maybe in 30 years we'll ask the same question about high school, but I claim that's not today's problem. Today's problem is the need to choose a system, each of which has its own idiosyncracies, and you aren't sure that the proof is reproducible in another. Also, you are not sure that the skills are transferable. Some problem with Computer Algebra, versus Maple/Mathematica. But I had this — I learned Pascal and Lisp, and was then expected to know how to program. The Lisp teacher told me to forget all I had learned. But these days we try to teach features of programming languages. Furthermore, we have a theorem that both Java and Python (say) are Turing-complete.

There is no equivalent of Turing-complete for formal logic systems. But what we should do is express theories as sets of axioms, and then translate where the axiom systems are compatible. And knows that an AC-proof, say. can't always be translated.

1.1.1 Logical Framework

Predicate Logic was the first framework (1928, Hilbert and Ackermann). But type theory [WR10] was not so expressible. Also [Church1940] was not so expressible. Then Martin-Löf. Five reasons for this.

1. No bound variables.
2. Nosyntax of proofs
3. No notion of computation.
4. No good notion of proof reduction.
5. Classical but not contructive.

Various solutions. Claims $\lambda\Pi$-calculus modulo theories is the uniform generalisation, see Dedukti system.

Very weak: term and proposition have to be defined. Shows Curry–de Bruijn–Howard correspondence, where Proof : $hboxProp \to$ Type. Then needs a specific rule to express this. Can get dependent implication as Coq, predicate subtyping (as in PVS) etc. We have 38 axioms. 19 of these just buidl predicate logic from Deduki's base. Complex Venn diagram about which axioms define which useful theory.

[Thiré] anaysed all the arithmetic library of MATITA, and gives first prove of Fermat's Little Theorem in Constructive Type Theory (therefore weaker than any existing proof). This one proof has been checked in several (?7) different systems. Currently have a Git repository "nubo" (cloud in Esperanto). Obervation that mosty users only use a small subset of the very powerful systems. Note how the libraries are much more similar than the systems.

Q Linear logic?

A We have all the structural rules. Hence we have too much power. Need a sub-structural system (same problem as LF has). But I think this is not needed for undergraduate mathematics.

1.2 Automated Grading of Automata inACL2s:Ankit Kumar

Look at feedback delays in manual grading. Hence advantages of autograders.

Example 1 (DSA) *Professor supplies specifcation/his solution. Student submits. First gets feedback "not a DSA" with explanation, e.g. "domain of transition function is nto of right type". Then "wrong alphabet". The "the following words are mis-classified".*

Example 2 (Turing machines) *As far as JHD could tell, the instructor provides test words for the final check.*

With multiple feedback allowed, 95% got full marks, whereas less than 20% did on manual grading. Feedback was positive. System is implemented in `gradescope.com` as a Docker image with ACL2 and ACL2s.

1.2.1 Q&A

Q Context?

A Class of c. 50; done by students working at home (Covid).

1.3 Automated Instantiation of Control Flow Tracing Exercises: Eisenhofer

Context; Introductory Java programming at TU Vienna. Lab claases, programming examination (90 minutes), and autograded online tests: consider this code: what should be the output/what should be added/.... Done in the Moodle framework. But the question pool is limited, and it's difficult to write new ones of similar difficulty. examples of how changing constants can change difficulty.

Our tool transforms given ranges of variables into an eexample, checked with an SMT-solver. We support a subset of Java (not OO). Integer arrays (experimental), string and string arrays. Most control flow. Also an ASSERT statement. Each state is represented by its own SMT state `x[@version][@writecount]`.

Chapter 2

12 July

2.1

We are based on "Array-based systems": $\mathcal{T}_I$ is the index theory, $\mathcal{T}_I$ is the element theory, then an array theory. We havea goal Φ which shoul dbe true in every reachable state. We first do goal strengthening, then parameter abstraction, and check. But a counterexample to the abstraction might be genuine, or might lead us to more goal strengthening.

Chapter 3

14 July

3.1 Collatz Conjecture: Emre Yolcu

Full version in [YAH21]. See 4-colour, Robbens, Keplet (Hale), Boolean Pythagiean triples for things proved by computer. So try Collatz.

String rewrite systems. Note definitions of "termination". One proof is to find a well-founded order which is decreasing. Or we can find an interpretation $s \mapsto [s]$ and get termination of interpretations. This, for example, handles "length decreasing". We map s to $[s](x) = M_s x + v_s$ extended compositionally, with $x \in \mathbf{N}^d$. Consider $\{aa \to bc, bb \to ac, cc \to ?\}$ and this was first (and only) proved by matrix interpretations. [Zantema2005] has a rewriting system on unary representations of $\mathbf{N}$ which is equivalent for Collatz. We could prove that this system has no terminating matrix interpretation, for any d. Consider $n \mapsto 3n/2$ if n even. Trivially terminating but no automated proof known. Can view binary as $\triangleleft(x) = 1lf(x) = 2x; t(x) = 2x + 1$, so 12=1100 is $f(f(t(\triangleleft(x))))$. Then introduces a mixed binary/ternary representation, adding $\{0(x) = 3x, 1(x) = 3x + 1, 2(x) = 3x + 2\}$. Can no longer chec for even by local transformations, but there are rewrite rules that move binary "to the left" until this happens. Then have 11 rules: 2 for Collatz and 9 for binary/ternary arithnetic. Still can't prove Collatz, but can do [Farkas2005]. All 11 subproblems with 10 of the 11 rules are soluble.

Any non-convergent trajectory cannot avoid $\{2, 3, 4, 6\}$ modulo 8.

Also looked at Mahler's 3/2 problem: $\exists ?\xi : frac(\xi \left(\frac{3}{2}\right)^k) < \frac{1}{2} \forall k$.

3.2 Verified Interactive Computation of Definite Integrals: Bohua Zhan

Shows examples from [GR07] (but these are indefinite!). Example: $\int_0^\pi \sqrt{1 + \cos(2x)} = \int_0^\pi \sqrt{1 + \cos^2 x - \sin^2(x)} = \int_0^\pi \sqrt{2\cos^2(x)} = \int_0^\pi \sqrt{2}\cos x = \sqrt{2}\sin(\pi) - \sqrt{2}\sin(0) =$

0 since we did $\sqrt{z^2} \mapsto z$ rather than $|z|$. An algebra system (SymPy 15.2) gets this wrong. [Duranetal2014] on correctness of computer algebra.

So what about using a proof system: not good at this, and not immediate. We aim to let users work in A CA system and convent these into HOL proofs. Our system is on HolPy. We have some simple inetgration rules (he mentioned Slagle).

$$\int_a^b f(x(x))g'(x)dx \Rightarrow \int_{g(a)}^{g(b)} f(u)du$$

and its converse are included, with f and g as parameters. Always report the trigonometic rules applied. Rule for spliting integrals at $c \in (a, b)$. Regards Slagle as a tree of and/or nodes. This gives us a sequence of steps (with the rationale logged) which can be checked in HolPy.

Inequality checking is a major part, e.g. simplifying $\sqrt{f(x)^2} \mapsto f(x)$ requires $f(x) \geq 0$. We have a heuristic version of this. Has an evaluation on various sets, including examination preparation books, MIT Integration Bee.

Q–Tim Daly What is your definition of "definite integration".

A Unclear

Q–LP Great. But Slagle's work [Sla61] is outdated, what about Rubi [JR10]?

A Like to extend

See [XLZ21].

Chapter 4

15 July

Further discussion with Tim Daly and authors of [XLZ21].

Q–TD I am looking t prove the algorithms correct. Possibly in Axiom/Lean

A

4.1 Automated Mathematician

[PoluSutskever2020], where the neural system found a neater proof than the manual one.

4.1.1 Free-form conjecturing

[Rabeetal2021a] More precisley goal-directed. Filtered out know, unprovable and "not useful". Done, apparently, with a transformer architecture. Lots of papers on transformers for formulae. Do we need to model the tree structure? See [LampleCharlton2020a]. "Out-of-the=box" works surprisingly well.

Chapter 5

16 July: ARCADE

5.1 Plaisted

?? depends on Classification FSG.

Gentzen allows cut

Collection $\Sigma^{\vdash}$ of inference rules and a computable function Σ^* such that Σ^* takes sets S of clauses to sequences of clauses, which is essentailly the proof. So DPLL might be one such. Write $\Sigma(S)$ for $\Sigma^*(S)$. Write $|\Sigma(S)|$ for the least length of a proof. Can't give a recursive bound for this. But we'd like to describe complexity somehow.

So let $||S||_H$ is the size of the smallest Herbrand set for S, and we can bound $|\Sigma(S)|$ in terms of $||S||_H$. [Pla15] does this for $O(H^{2^{H^2}})$, where $H := ||S||_H$. But redoing it gives differnet versions $(2H|S|)^{2^{2H^2}}$ — still doubly exponential, but this I have confidence in. This is for resolution.

DPLL might be $O(\ldots)$ (simply exponential): which might be better. But you never need a clause larger than H after factoring. The number of such clauses is at most $(c+H)^H$ after factoring (up to H variables), possibly $(c+2H)^H$ with DAG pointers. We can bound things like sum of clause sizes. Complexity isn't whole answer: look at Theory/Practice for SAT.

5.1.1 Q&A

Q Nice. People with look at constructing examples.

A Thanks.

Q Thanks - can this show that an algorithm is "good"? It has been hard for implementers to get published. But producing such a proof would take as long

A

Q–Martin Suda So you ask the cost of refuting after I have found the right clauses? So this is the propositional complexity?

A Yes, but some proofs have enormous H. The theorem prover doesn't know H.

Q How does this compare to proor complexity for propositional systems.

A This has various P/NP issues. Exponential is the worst bound we have here.

Q Isn't this the same at the propositional level. People argue in terms of implementations.

A Possibly.

5.2 JHD

See slides at `https://staff.bath.ac.uk/masjhd/Slides/JHDatARCADE2021.pdf`.

5.2.1 Q&A

Q–PF There are other methods, such as Interval Constraint Propagation.

A Indeed, and the proof of the first example in [ADE+20] (non-overlapping circles) is essentially an Interval Constraint Propagation proof. But I am not sure about the second one (overlapping but not intersecting circles).

PF I think this could also be done by Interval Constraint Propagation. [Later] look at [FHT+07] as a starting point.

Q–Platzer This is super-important work. Later:

> I really enjoyed your ARCADE talk and couldn't agree more with your conclusion how important it is to get trustworthy correct nonlinear real arithmetic decisions by verification.
>
> We use QE especially for decision problems and often but not always even purely universal ones every day in our KeYmaera X theorem prover for verifying cyber-physical systems such as cars, aircraft, etc. Its uses of real arithmetic decision procedures need to be trustworthy.
>
> You may be interested in John Harrison's verified certificates based ultimately on the Positivstellensatz: [Har07].
>
> We've also been working on related certificate generation and checking based on Gröbner bases and the real Nullstellensatz, which can be checked in fairly easy ways in KeYmaera X: [PQR09].
>
> Of course, the downside is the need to generate those certificates. That's why we've also looked at static verification giving an

implementation with a correctness proof once and for all. My student Katherine, who is CCed, is doing her PhD on formally verified nonlinear real arithmetic QE or decision procedures.

For the univariate fragment of the Ben-Or Reif Kozen algorithm, she developed and verified correct an implementation in Isabelle/HOL: [CTP21].

Katherine also developed a verified implementation of quadratic virtual substitution in Isabelle/HOL, which ends up being quite practical: [SCMP21]. Her verified virtual substitution implementation also gives correct results when only some of the variables can be eliminated, which may give you a way of integrating it into your CAD-based algorithm without needing to generate proof certificates.

Now none of those are complete, but at least we hope to have a credible route to achieving a good tradeoff of verifiability and performance in real arithmetic.

A Thanks - will investigate.

5.3 Evaluation of ATPs: Michael Rawson

Run your system, on some problems, and count how many you've solved. Pick the problem set (there are many: TPTP, MPTP2078, Mizar40 etc.). Cherry pick some fragment. Quotes very reminiscent (to JHD) of [McD81]. Reproducibility is a real problem. Various referee's comments analysed. A load of tables that no-one understands. "Path to the Ideas Scrapheap". Any evaluation method should be simple. Vampire at least exposes magic numbers, rather than baking them in.

Also, what about alternative forms of progress (simplicity etc.)?

Can we use emulation for reproducible runs? Can we check speed by random sampling? Is there evaluation by fast portfolio creation (work at Manchester). Case study approaches, and leaving evaluation to competitions. Do we want a living document, describing approaches etc.

5.3.1 Q&A

Q–Giles We also discussed the social aspect. Also usability is more important than performance for many people.

A Agreed.

Q–JHD Have you looked at [McD81]?

A Not yet!

5.4 Relevance and Abstraction: Plaisted

Various examples, e.g. Marc Fuchs on Lemmas, [Pud07] on semantic selection of Premises. We can try deleting clauses in turn. This will give you a local minimum.

Better idea might be semantics abstractions.

Bibliography

[ADE+20] E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z.P. Tonks. New Opportunities for the Formal Proof of Computational Real Geometry? *SC^2'20: Fifth International Workshop on Satisfiability Checking and Symbolic Computation CEUR Workshop Proceedings*, 2752:178–188, 2020.

[CTP21] Katherine Cordwell, Yong Kiam Tan, and André Platzer. A Verified Decision Procedure for Univariate Real Arithmetic with the BKR Algorithm. In Liron Cohen and Cezary Kaliszyk, editors, *12th International Conference on Interactive Theorem Proving (ITP 2021)*, volume 193 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/13909`, `doi:10.4230/LIPIcs.ITP.2021.14`.

[FHT+07] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, 1(3-4):209–236, 2007. URL: `http://jsat.ewi.tudelft.nl/content/volume1/JSAT1_11_Fraenzle.pdf`.

[GR07] I.S. Gradshteyn and I.M. Ryzhik. *Table of Integrals, Series and Products* 7th edition (ed. A. Jeffrey and D. Zwillinger). Academic Press, 2007.

[Har07] J. Harrison. Verifying Nonlinear Real Formulas Via Sums of Squares. *International Conference on Theorem Proving in Higher Order Logics TPHOLs 2007*, pages 102–118, 2007.

[JR10] D.J. Jeffrey and A.D. Rich. Reducing Expression Size Using Rule-Based Integration. In S. Autexier *et al.*, editor, *Proceedings CICM 2010*, pages 234–246, 2010.

[McD81] D. McDermott. Artificial Intelligence Meets Natural Stupidity. In J. Haugeland, editor, *Mind Design*, pages 143–160. 1981.

[Pla15] D.A. Plaisted. History and prospects for first-order automated deduction. *In International Conference on Automated Deduction*, pages 3–28, 2015.

[PQR09] A. Platzer, J.-D. Quesel, and P. Rümmer. Real World Verification. In R.A. Schmidt, editor, *Proceedings CADE 2009*, pages 485–501, 2009.

[PS21] A. Platzer and G. Sutcliffe, editors. *Automated Deduction — CADE 28*, volume 12699 of *Springer Lecture Notes in Computer Science book series*, 2021.

[Pud07] P. Pudlák. Semantic Selection of Premisses for Automated Theorem Proving. *CEUR-SW*, 257:4:1–4:18, 2007.

[SCMP21] M. Scharager, K. Cordwell, S. Mitsch, and A. Platzer. Verified Quadratic Virtual Substitution for Real Arithmetic. `https://arxiv.org/abs/2105.14183`, 2021.

[Sla61] J. Slagle. *A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus*. PhD thesis, M.I.T., 1961.

[WR10] A.N. Whitehead and B. Russell. Principia Mathematica. *Cambridge University Press*, 1910.

[XLZ21] Runqing Xu, Liming Li, and Bohua Zhan. Verified Interactive Computation of Definite Integrals. In Platzer and Sutcliffe [PS21], pages 485–503.

[YAH21] Emre Yolcu, Scott Aaronson, and Marijn J. H. Heule. An Automated Approach to the Collatz Conjecture. In Platzer and Sutcliffe [PS21], pages 468–484.