CS6180 Lecture 26 – Automated Reasoning and Ultra-Intuitionism

Robert L. Constable

Abstract

This lecture will briefly consider two topics. The main one is the relationship between *automated reasoning*, an AI topic, and constructive type theory. The other topic is a brief mention of a radical approach to the foundations of mathematics called *ultra-intuitionism* and a discussion of *impredicativity*, a feature of logics considered dangerous and even unacceptable to some excellent mathematicians and logicians. Ultra-intuitionism denies the existence of infinite types or sets, accordingly mathematics is concerned with constructive reasoning about relatively small finite types (less than 10^{12} elements).

1 Automated Reasoning

The study of automated reasoning originated in Artificial Intelligence (AI) [29], and it continues to have strong ties to AI through the increasing use of *proof assistants* and the efforts to make them "smarter".[1] Proof assistants such as Agda, Coq, Nuprl, and Lean are designed to help users create formal proofs and formally correct theorems and software that provably meets precise specifications.

Recent applications of proof assistants have focused on creating provably correct software as featured in the *Software Foundations Project* [31]. There is direct and *immediate economic value* to this activity, and it is thus possible to find support from industry as well as from NSF and other science funding agencies.[2] It is assumed that some of the funding for applied work will enhance the effectiveness of proof assistants. To some extent this is born out in practice.

Other applications that have gathered attention are uses of proof assistants to check solutions to famous problems in mathematics such as the Four Color Theorem [16] and a machine checked proof of the Odd Order Theorem in group theory [17]. Recently Hales and colleagues checked his proof of the Kepler Conjecture using HOL [13]. There is other high profile work using proof assistants in mathematics, especially in formalizing elements of homotopy theory [33, 34] and in applications to *cyber physical systems* [3]. We have seen an example of this with the results of Dr. Bickford and his collaborators Coquand and Anders as they confirmed the *constructive validity* of the *Univalence Axiom* in homotopy theory.

Steven Pinker's book, *How the Mind Works* [32], provides an engaging glimpse of the AI side of automated reasoning. He discusses a "computational theory of mind" and compares the results of

[1] Some historians credit a summer meeting at Cornell in 1957 as the beginning of AI. The paper by Newell, Shaw, and Simon cited above was presented at that meeting.

[2] Some software companies have paid consulting firms to evaluate the cost effectiveness of this practice, and the results have so far confirmed its value.

computer science on this topic with those of other approaches to understanding the mind. One of his comments is precisely this:" *Now what is wrong with this picture? The philosophers are accusing the computer scientists of fuzzy thinking.*"

In a 1998 long (103 page) article on type theory [10], I make a case for the value of AI in proof assistants. The early articles in this area inspired me and my PhD students, in particular articles of McCarthy, Newell, Shaw, and Simon [23, 24, 29, 4]. These articles led to the work of Robin Milner at Edinburgh [18] that changed life in Cornell CS in both PL and AI. Robin Milner also provided a second academic home for what would become Nuprl. At Edinburgh I also became familiar with the work of Boyer and Moore [4, 5, 6] and the work of Alan Bundy [7] on the *Clam-Oyster* system. This AI group liked to say that their system had a "PRL" in it.

Another major influence on the Nuprl logic was the *Automath* system of N.G. de Bruijn [11, 12, 21, 27, 35]. This was a completely formal system for doing all of mathematics. However, it did not employ any automated reasoning tools. De Bruijn was in the mathematics department and his university did not have a computer science department. We learned how hard it was to do formal proofs without help from a system, and this led us to embrace as much AI technology as we could bring to bear. The work of Robin Milner on LCF gave us the tools we needed.

The lecture will give examples of various milestones in our efforts include AI results and techniques into the Nuprl proof assistant. These advances made it possible to formalize significant elements of mathematics and to solve open problems in mathematics [20, 26, 8] – the last one open for sixty seven years.

1.1 LCF tactics and tacticals

In the *Edinburgh LCF* proof assistant [18], automation of reasoning was accomplished by writing *tactics* that decomposed a goal into subgoals and tacticals which combine tactics to create larger ones. Here is an example of a simple LCF tactic.

SIMPTAC : tactic $(w, ss, w1)| - -- > [(w', ss, w10)]$ where w' is the simplification of w by applying the rule ss.

Tacticals are methods of combining tactics. For example, $THEN$ is a tactical that can sequence tactics. Here is what the LCF manual says about it.

The tactic T1 THEN T2 first applies T1 to the goal, then applies T2 to all of the resulting subgoals. Failure of either T1 or T2 fails.

The tactic T1 ORELSE T2 acts on a goal as T1 unless the later fails, in which case it acts as T2.

The tactic REPEAT T applies T repeatedly to the goal and all subgoals so produced, until T is no longer applicable (i.e. would fail). REPEAT T never fails. The LCF definition is this:

letrec REPEAT(T) g = ((T THEN REPEAT(T)) ORELSE IDTAC) g ;;

1.2 Formal Digital Library (FDL)

The Nuprl system organizes all of its definitions, theorems, and tactics into a *database* that we call the FDL (Formal Digital Library) or just the *Library* [2, 9, 1, 22]. This is more than a collection of "books" or "theories." The library is shared by all Nuprl users, and can be used to coordinate users as they collaborate on building a theory or creating a proof. In addition, the FDL can operate autonomously to some extent by running various reasoning tactics in the background checking for computational equality of terms or for subtyping relations or by proving results in decidable theories, especially in numerical theories.

The library provides many utilities for searching, and it can be accessed concurrently by many processes and many users. There are features that allow users to cooperate in creating a proof. The FDL can also operate in an *autonomous mode* where it explores proof possibilities on its own in the background. It has happened many times that Nuprl will find a proof or make intelligent suggestions that lead to novel proofs.

Here is a list of some of the ways the Nuprl FDL assists users in developing proofs and theories.

1. Enable multiple users to work on a proof together.
2. Allow multiple refiners to be working on the same proof, *asynchronous refinement*.
3. Automatically find "similar results" in the Library to a current goal.
4. Set *default tactics* to run automatically after various kinds of proof steps.
5. Offer suggestions for the next step in a proof, "here is what I would try next."
6. Use machine learning to find patterns of inference and automatically explore them in the background.

One question that might arise in the near term is whether Nuprl should be given explicit credit for a proof. Acknowledging it the first time will require a "high bar" to avoid ridicule, but in due course this will happen, and the chances are good that it will happen with Nuprl.

We are preparing the FDL for use with various *machine learning tools*. We are confident that before long we will be able to exhibit this capability in a compelling way. Such results will attract significant attention and funding, and they will introduce a *new era in mathematical discovery.*

2 Ultra-intuitionism and Predicative Mathematics

For some mathematicians, the intuitionistic restrictions are not sufficient to create a completely safe and sound basis for mathematical truth. The mathematician A.S. Yessenin-Volpin does not believe that the natural number 10^{12} exists. He says that no one can count that far – at one number per second, it would take more than 20,000 years to reach this number. We might admit that we can't count this high, but we seem to grasp the number.

2.1 Ultra-intuitionism

Ultra-intuitionism is a radical restriction of mathematics.

2.2 Predicative versus Impredicative Mathematics

There are more mathematicians who would be flexible on the size of numbers yet who object to definitions that are *impredicative*. These are definitions that quantify over the set being defined in its definition. When we examine the standard axioms for *Peano Arithmetic* (PA) [30] or the intuitionistic axioms for *Heyting Arithmetic* (HA) [19] we see quantification over the type (or set) of natural numbers $\mathbb{N}$. This makes the account of arithmetic impredicative. The Princeton mathematician gives a predicative account of $\mathbb{N}$ in his book Predicative Arithmetic [28]. He caught the attention of the mathematical world when he presented a proof of the inconsistency of Primitive Recursive Arithmetic (PRA) [**?**]. Later Terence Tao found an error in his proof. Nelson was interested in using the QED system to check his proof of inconsistency, but the status of this work is unclear.

There are a larger number of excellent mathematicians who prefer *predicative* mathematics [14, 15, 28]. In the earliest days of constructive type theory, we implemented an impredicative recursive type [25], but we found that we could use Brouwer's Bar Induction to cover the important topics, so we no longer use impredicative recursive types. Nevertheless, some mathematicians such as Edward Nelson of Princeton University, would say that *even the number theory of constructive type theory is impredicative* [28]. We will briefly discuss this viewpoint.

We can examine a simple example of impredicative reasoning about numbers, showing that *ordinary mathematical induction is impredicative.* This example by Edward Nelson makes the point very succinctly. It is included as supplementary reading for this lecture, copied from the book *Predicative Arithmetic* [28].

References

[1] Stuart Allen, Mark Bickford, Robert Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and Evan Moran. Innovations in computational type theory using Nuprl. *Journal of Applied Logic*, 4(4):428–469, 2006.

[2] Stuart F. Allen, Robert L. Constable, and Lori Lorigo. Using formal reference to enhance authority and integrity in online mathematical texts. *Journal of Electronic Publishing*, 2006.

[3] Jacob Aron. Beyond knowledge. *New Scientist*, pages 28 –31, 2015.

[4] R. S. Boyer and J. S. Moore. *A Computational Logic*. Academic Press, New York, 1979.

[5] R. S. Boyer and J. S. Moore. *A Computational Logic Handbook*. Academic Press, 1988.

[6] Robert S. Boyer and J. Strother Moore. Integrating decision procedures into heuristic theorem provers: A case study with linear arithmetic. In J. E. Hayes, D. Michie, and J. Richards, editors, *Machine Intelligence*, volume 11, pages 83–124. Clarendon Press, 1988.

[7] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In Mark E. Stickel, editor, *Proceedings of the 10^{th} International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 647–648. Springer-Verlag, 1990.

[8] Robert Constable and Mark Bickford. Intuitionistic Completeness of First-Order Logic. *Annals of Pure and Applied Logic*, 165(1):164–198, January 2014.

[9] Robert Constable, Richard Eaton, Jon Kleinberg, and Lori Lorigo. A graph-based approach towards discerning inherent structures in a digital library of formal mathematics. 3119:220–235, 2004.

[10] Robert L. Constable. Types in logic, mathematics and programming. In S. R. Buss, editor, *Handbook of Proof Theory*, chapter X, pages 683–786. Elsevier Science B.V., 1998.

[11] N. G. de Bruijn. The mathematical language Automath: its usage and some of its extensions. In J. P. Seldin and J. R. Hindley, editors, *Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, pages 29–61. Springer-Verlag, 1970.

[12] N. G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In R. P. Nederpelt, J. H. Geuvers, and R. C. De Vrijer, editors, *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*, pages 865–935. Elsevier, Amsterdam, 1994.

[13] Halles et al. A formal proof of the kepler conjecture. *arXiv:1501*, pages 1 – 21, 2015.

[14] Solomon Feferman. A language and axioms for explicit mathematics. In J. N. Crossley, editor, *Algebra and Logic*, volume 480 of *Lecture Notes in Mathematics*, pages 87–139. Springer, Berlin, 1975.

[15] Solomon Feferman. Predicativity. In Stewart Shapiro, editor, *The Oxford Handbook of Philosophy of Mathematics and Logic*, pages 590–624. Oxford University Press, Oxford, 2005.

[16] Georges Gonthier. Formal proof – the Four Color Theorem. *Notices of the American Math Society*, 55:1382–1392, 2008.

[17] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-checked Proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2013.

[18] Michael Gordon, Robin Milner, and Christopher Wadsworth. *Edinburgh LCF: a mechanized logic of computation*, volume 78 of *Lecture Notes in Computer Science*. Springer-Verlag, NY, 1979.

[19] A. Heyting. *Intuitionism, An Introduction*. North-Holland, Amsterdam, 1966.

[20] Douglas J. Howe. The computational behaviour of Girard's paradox. In D. Gries, editor, *Proceedings of the 2^{nd} IEEE Symposium on Logic in Computer Science*, pages 205–214. IEEE Computer Society Press, June 1987.

[21] L.S. Jutting. *Checking Landau's 'Grundlagen' in the AUTOMATH System*. PhD thesis, Math Centre, Amsterdam, 1979.

[22] Lori Lorigo. *Information Management in the Service of Knowledge and Discovery.* PhD thesis, Cornell University, 2006.

[23] J. McCarthy. Computer programs for checking mathematical proofs. In *Proceedings of the Symposium in Pure Math, Recursive Function Theory*, volume V, pages 219–228. AMS, Providence, RI, 1962.

[24] J. McCarthy. A basis for a mathematical theory of computation. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 33–70. North-Holland, Amsterdam, 1963.

[25] P.F. Mendler. *Inductive Definition in Type Theory.* PhD thesis, Cornell University, Ithaca, NY, 1988.

[26] Chetan Murthy. An evaluation semantics for classical proofs. In *Proceedings of the 6^{th} Symposium on Logic in Computer Science*, pages 96–109, Vrije University, Amsterdam, The Netherlands, July 1991. IEEE Computer Society Press.

[27] R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and The Foundations of Mathematics.* Elsevier, Amsterdam, 1994.

[28] Edward Nelson. *Predicative Arithmetic.* Princeton University Press, 1986.

[29] A. Newell, J.C. Shaw, and H.A. Simon. Empirical explorations with the logic theory machine: A case study in heuristics. In *Proceedings West Joint Computer Conference*, pages 218–239, 1957.

[30] G. Peano. *Selected Works.* University of Toronto Press, Toronto, 1973. Edited by H.C. Kennedy.

[31] Benjamin C. Pierce, Chris Casinghino, Michael Greenberg, Vilhelm Sjberg, and Brent Yorgey. *Software Foundations.* Electronic, 2013.

[32] Steven Pinker. *How the Mind Works.* W.W. Morton, New York, 1997.

[33] Univalent Foundations Program. *Homotopy Type Theory.* Univalent Foundations Program, 2013.

[34] Vladimir Voevodsky. A C-system defined by a universe category. arXiv 1409.7925, 2015.

[35] Freek Wiedijk. A contemporary implementation of Automath. Talk presented at the Workshop on 35 years of Automath, Heriot-Watt University, Edinburgh, Scotland, April 10–13, 2002.