Automated Reasoning: From Bold Dreams to Computer Science Methodology

Robert L. Constable

Cornell University

From Bold Dreams to Computer Science Methodology

The title suggests my main point -- that our subject, automated reasoning, is integral to the basic methodology of computer science.

Outline

My plan is to develop this theme.

-- first by connecting it to Herbrand's work

-- then by looking at why automated reasoning is becoming so important in CS

-- finally by looking at how other Herbrand award scientists have contributed to my work and how that work advances this trend in CS

Computer Science

A definition of CS that I like is:

Computer science is the automation of intellectual processes.

It is a widely used definition in many areas of computer science.

Nature of Computer Science

Although Herbrand did not know of computer science and did not have a vision like that of Leibniz for a calculus of reason (*calculus ratiocinator*), I think he would have liked CS and this definition very much.

Here is why I think this.

Herbrand and Computer Science

Claude Chevalley said this about Jacques Herbrand (almost verbatim):

"Just as mathematical physics permits us to penetrate further and further into the structure of matter, logic allows us to describe something nearer yet to us than our sensations: our intellectual thought."

Herbrand and CS

Nowadays this quote could have been:

" Just as mathematical physics permits us to penetrate further and further into the structure of matter, logic and computer science allow us to precisely describe something nearer yet to us than our sensations: our intellectual thought."

Constructivity in CS

Computer Science is one of the most mathematical disciplines, and much of its mathematics is about how to find efficient algorithms and good data structures for them.

So computer scientists understand constructive mathematics intuitively.

Most mathematicians also prefer constructive proofs if they can find them.

Constructivity in CS

I have always been very interested in constructive logic because constructive formal theories are programming languages.

In this setting the computational content of constructive proofs are programs and elements of data types.

Herbrand's Constructive Approach

Warren Goldfarb edited the 1968 book

Jacques Herbrand: *Écripts Logiques*

in which he says about Herbrand's Theorem:

"… this theorem also furnishes constructive insight into nonconstructive notions such as satisfiability."

Constructive Methods

Herbrand used constructive (finitist) methods and could not bring himself to state what became Gödel's completeness theorem.

See Martin Davis, *Prehistory and Early History of Automated Deduction, in Automation of Reasoning 1:*

"… Herbrand could not permit himself the non-constructive step necessary to obtain the completeness theorem itself."

Constructive Methods

I think Herbrand would have liked constructive type theory and the approach of Kolmogorov and Heyting in axiomatizing constructive logic.

He might even have discovered first the completeness result for intuitionistic first-order logic that we can now prove constructively.

Outline

-- then by looking at why automated reasoning is becoming so important in CS

-- finally by looking at how other Herbrand award scientists have contributed to my work and how that work advances this trend in CS

Automated Reasoning is Central

On this view of computer science, automated reasoning is essential. That is not yet the common narrative, except for people who have experienced the power of proof assistants and other automated reasoning tools.

This Understanding Spreads

In computer science, automated reasoning was associated first with Artificial Intelligence. Now it applies in Programming Languages, and that subfield is spreading our ideas and proof technology into Systems.

Soon automated reasoning could be embraced in Computing Theory and then in computer science as a whole, think of Probabilistically Checkable Proofs (PCP) for instance, versus incompleteness and undecidability.

Step by Step

- AI and Logic made AR into a field.
- AR changed the PL field – proof assistants.
- PL became a vector for AR into Systems.
- Systems is a game changing area of CS.
- Systems needs AR to fly safely in the clouds.
- Theory, like math, will use AR for results.

PL as a Vector for AR into Systems

Coq and Nuprl are new kinds of programming assistants, supporting rich dependently typed programming languages.

Programmers learn to specify their code with dependent types and logic.

They learn to prove that programs, protocols, and systems meet specifications, and test those specifications against behaviors and models.

Formal proofs are done with AR

I can speak with some authority about Coq and Nuprl proofs, the PRL Group uses both proof assistants.

We use Coq to formalize the constructive type theory native to Nuprl, CTT14 [Anand, Rahli 14],

Towards a Formally Verified Proof Assistant, ITP 14.

We use Coq to formally prove our rules correct semantically and to export CTT14, the type theory implemented by Nuprl.

Outline

-- finally by looking at how other Herbrand award scientists have contributed to my work and how that work advances this trend in CS

Coq and Nuprl Proof Assistants use Advanced AR

1. Substantial induction packages informed by Boyer & Moore, Alan Bundy.
2. Proof methods for non-classical logics using results of Fitting.
3. Bibel gave us the connection method and Christoph Kreitz, a long term partner who with Otten gave us JProver for MetaPRL, also used by Coq at one point.
4. Clarke style model checkers are used at specification time and during proof attempts.

Coq and Nuprl Proof Systems use Advanced AR

5. Andrews and de Bruijn made us brave about automating type theory reasoning.
6. Kapur, Dershowitz, Huet, and Paulson gave Nuprl a remarkable rewrite system implemented by Paul Jackson and extended regularly by Mark Bickford.
7. Everyone uses unification from Robinson and the Davis, Putnam procedure.
8. We use cong closure of Nelson, Kozen, et al.

Coq and Nuprl Proof Systems use Advanced AR

Nearly half of the Herbrand awards represent contributions to AR driven proof assistants that I know well such as Coq, HOL, Nuprl, and MetaPRL.

These provers also use the Edinburgh LCF tactic mechanism that I associate with Robin Milner's Turing Award. These systems added automated techniques in the LCF setting right from the start and proved some of them correct.

Conclusions

It is a near certainty that proof assistants like Agda, Coq, HOL, Nuprl, and MetaPRL will continue to advance mathematics, e.g. Homotopy Type Theory is a current example.

They will support formal methods critical to security, reliability, and trust – especially in cloud based computing.

Conclusions

Noteworthy milestones accumulate such as seL4 verification, Multi-Paxos synthesis, formal models of C and Java, the Four Color Theorem and Feit-Thompson Theorem (odd order theorem), solutions to open problems, automatic conversion of classical results to constructive ones, and so forth.

We will find measures like the older de Bruijn index, measures of trust, fun factors and stun factors.

Predictions

Everyone who uses proof assistants senses their value in CS and Maths education as well as in programming.

Something revolutionary will happen when proof assistants reach the schools, as they will in due course.

What I Believe

I believe that Automated Reasoning will profoundly change the way mathematics and programming are done and taught, the way software is built, and the way logic and computer science are understood.

Special Thanks to My Students

The work cited in the award could not have been done without my many superb PhD students who worked with me in the area, nearly half of my students including these:

M. O'Donnell, J. Bates, R. Harper, S. Allen,
N. Mendler, D. Howe, T. Griffin, S. Smith, D. Basin,
R. Cleaveland, P. Jackson, R. Moten, C. Murthy,
K. Crary, J. Underwood, J. Caldwell, L. Lorigo,
J. Hickey, A. Nogin, A. Kopylov, A. Anand.

Others as Well

Ed Clarke worked on AR after he graduated, and Kurt Mehlhorn only recently has become interested in the area.

Many of my Cornell colleagues have also contributed and have been very supportive, especially Dexter Kozen whom I mentioned already for his important AR work used in Nuprl.

Others as Well

My long time collaborator Mark Bickford and our team research staff, Richard Eaton and Vincent Rahli, have been the heart of Nuprl and our recent work on distributed systems with the Cornell systems group, especially with Robbert van Renesse and Ken Birman.

References

- Developing Correctly Replicated Databases Using Formal Tools, Nicolas Schiper, Vincent Rahli, Robbert van Renesse, Mark Bickford, and Robert L. Constable, *The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN),* Atlanta, GA, June 2014.

- Towards a Formally Verified Proof Assistant, Abhishek Anand, Vincent Rahli, *The 5th Conference on Interactive Theorem Proving (ITP),* Vienna, Austria, July 2014.

- A Type Theory with Partial Equivalence Relations as Types, Abhishek Anand, Mark Bickford, Robert Constable, Vincent Rahli, *TYPES 2014: Types for Proofs and Programs*, Paris, May 2014.

THE END, THANK YOU

Notes and References

1. *Automation of Reasoning* Vols 1 and 2 edited by J. Siekmann, G. Wrightson, 1983.
2. Jacques Herbrand: *Écripts Logiques* edited by Warren Goldfarb 1968.