Mechanizing and Automating Mathematics: In honor of N.G. de Bruijn

Automath started in 1967 by N.G. de Bruijn. Automath (automating mathematics) was the first system to use computer technology to check the correctness of whole books of mathematics. During his work on Automath, N.G. de Bruijn discovered many concepts that still remain of great relevance to the theory and practice of computation. For example:

- De Bruijn indices [8] still play an important role in the implementation of programming languages and theorem provers.
- De Bruijn's new type systems were influential in the discovery of new powerful type systems [6].
- De Bruijn re-invented the Curry-Howard isomorphism (also referred to as the Curry-Howard-de Bruijn isomorphism). Independently of Curry and Feys [7] and Howard [9] , we find a variant of the propositions as types principle in the first Automath system of de Bruijn (AUT-68 [12, 4]). Though de Bruijn was probably influenced by Heyting (see [5] in [12], p. 211), his ideas arose independently from Curry, Feys and Howard. This can be clearly seen in Section 2.4 of [3], where propositions as types (or better: proofs as terms) is implemented in a way which differs from the method of Curry and Howard (see [10]).
- The Landau book [11] on the foundations of analysis remains the only fully encoded and checked mathematical book in any theorem prover. The Landau book was encoded in Automath in the seventies by Bert van Benthem-Jutting (who was a Ph.D. student of de Bruijn at that time) [1, 2].

Automath was written in Algol 60 and implemented on the primitive PCs of the sixties. Thirty five years on, both technology and theory have evolved very much leading to impressive new directions in using the computer for manipulating and checking mathematics. These advances had influence on many aspects of computational systems. To celebrate thirty five years of Automath and some of the impressive directions in using computers for mathematics, a workshop was held during 10–13 April 2002 at Heriot-Watt University, Edinburgh, Scotland (see http://www.cee.hw.ac.uk/~fairouz/automath2002/). This area is so huge now that it cannot be covered in one workshop. The workshop, although very successful, only covered a small fraction of the directions in this exciting field. The importance of formalizing, mechanizing and automating mathematics is now undisputable and much work is carried out all over the world on this fascinating topic.

It is beyond doubt that de Bruijn is greatly respected, deeply influential, and well liked by the leaders and researchers in the field. The evidence for this is the endless lessons and ideas that the field has adapted from his Automath. But another evidence for me is the fact that such a high profile community came together to celebrate 35 years of his Automath and at such an unbelievably short notice (The whole idea of the workshop was born from scratch at the end of January 2002). This is no surprise at all considering the man himself and his impressive Automath and ideas. It was so touching however, to see how everyone (including many people who could not attend due to already scheduled commitments but who sent many messages of praise for de Bruijn and his Automath) helped make this event a success for de Bruijn.

In addition to the invited talks at the Automath workshop in April 2002, there was also an interesting list of contributed talks selected by an efficient and competent programme committee that consisted of

- Mauricio Ayala-Rincón (Brasilia University, Brasil),
- Fairouz Kamareddine (chair) (Heriot-Watt University, UK)
- Rob Nederpelt (Eindhoven University of Technology, NL)
- Giovanni Sambin (University of Padova, IT)
- Joe Wells (Heriot-Watt University, UK)

Sparked by the workshop, a call for papers for a special issue on mechanizing and automating mathematics was made in February 2002. Twenty Six papers were received most of which were of high quality. Of those twenty six papers, eight were selected for this special issue. These papers include:

- The article of Bancerek, Rudnicki presents the effort of the Mizar team to formalize the contents of a well known modern text on lattice theory, "A Compendium of Continuous Lattices" (CCL), by G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott, Springer-Verlag, 1980. The formalization is carried out in the MIZAR system, as part of the even larger "MIZAR Mathematical Library" enterprise. The CCL project started in

1996 and involves a large group of people. The paper is accessible to readers with very little previous knowledge of MIZAR.

- The article of Bertot, Magaud and Zimmermann deals with a complete proof in the proof assistant Coq of a real algorithm to compute the square root of large integers. Three steps are studied:

    - A formal description of an effiecint version of the sqrt program.
    - The correctness of the implementation (written in C) where a description of the conditions under which the algorithm operates correctly is given. The difficulty resides in the use of imperative languages like C. However, the paper uses a Coq tool called Correctness which produces verification conditions that can be proved from an annotated program.
    - The correctness of the memory management where the formalization covers even the details of memory usage and pointer arithmetic. The paper shows that the same scheme can be used for the certification of the program and of the memory management.

- The article of Bezem, Hendriks and de Nivelle is concerned with translating refutation proofs obtained by resolution to Coq, using reflection and an encoding of the resolution proof in minimal logic. The paper adds automated proof search to type theory, more specifically the paper adds resolution to Coq making type systems usable in real applications. This leads to a system which has the following features:

    - A way of representing clauses that admit both a classical resolution strategy and an interpretation in classical dependent type-theory.
    - A representation of clauses in minimal logic such that the λ-representation of resolution steps is linear in the size of the premises. A naive translation would be exponential in the length of the resolution proof.
    - A translation of resolution proofs into lambda terms, yielding a verification procedure for those proofs.

- The article of Hendriks describes a formalization of natural deduction proofs for intuitionistic first order logic in the proof assistant Coq. Soundness of the formalization is proved by interpreting logic formulas into formulas in Coq. As an example, the author formalizes Prawitz' permutative conversions and proves the subject reduction property. This paper gives a complete (first-order) formalization of natural deduction for first-order logic using analytic judgements. Features of this work include a Coq implementation and a library which can be reused for several purposes such as the investigation of the meta-theory of deduction systems and the automation of proof-search of first-order theorems.

- The article of Nguyen, Kirchner and Kirchner describes an approach to incorporate term rewriting in proof assistants based on constructive type theory such as Coq. It contributes to the expanding field of hybrid theorem provers that try to combine strong points of different proof assistants in order to improve their performance and enhance their efficiency. The paper proposes a method to describe rewriting proofs in a calculus called ELAN, then to translate ELAN proofs to Coq proofs. The translation is proved to be sound, thus we can safely use rewriting techniques in ELAN to prove an equation in Coq. The idea is that the task of subgoals rewriting is delegated to ELAN. Then, ELAN proofs are sent back to Coq for verification. In order to carry out this movement between Coq and ELAN, the authors use a calculus $\rho\sigma$ which has explicit substitutions and explicit rewriting.

- The article of Prevosto and Doligez presents the programming language FOC, which is specifically designed to develop computer algebra systems. This language incorporates facilities to prove properties of the developed programs. Object-oriented style inheritance allows mathematical structures to be specified and extended in a natural way. After presenting the core set of features of FOC, the authors describe the static analysis, which rejects inconsistent programs. Static analysis can check properties of specifications which include information such as when proofs of theorems or termination proofs for functions can be inherited. The programs written in FOC are translated to OCAML to be compiled and the system Coq is used to verify the proofs.

- The article of Wiedijk resurrects Automath and is hence of historical value. The paper describes a new implementation (called 'aut') of de Bruijn's Zandleven Automath checker from the seventies. It describes in some detail the features of aut and was written to restore a damaged version of Jutting's translation [1] of Landau's book [11]. Wiedijk establishes that aut is quite fast, even when compared to current theorem prover systems (aut can check the translation of a full book in under a second). The implementation covers different Automath dialects. In addition to the implementation, Wiedijk discusses different aspects of the position of the type systems of Automath within modern type systems. For example, Wiedijk discusses the implications in Automath of having the type of a λ-expression be itself again a λ-expression. In a pure type system the type of a λ-expression is a *product type* or a Π-expression.

- The article of Wenzel and Wiedijk compares two declarative proof systems: Mizar and Isar. The paper highlights the main differences between Mizar and Isar. It begins with a literature review describing other attempts to imitate the Mizar system. Some similarities between Mizar and Isar are described. For the purpose of comparison, a well-known theorem is proved using both systems, namely Euclid's proof that there are infinitely many primes. Eighteen differences are presented which illustrate that:
    - Both systems have their strong points from a user's point of view (so it's not the case that one is generally "better" than the other).
    - Both systems can be improved by learning from the strong points of the other system.

I am very grateful to all of the referees for their highly valuable work. In particular, I am grateful to: Mark Aagaard, Peter Aczel, Thorsten Altenkirsch, Andrea Asperti, David Aspinall, Franco Barbanera, Andrej Bauer, Gilles Barthe, Stefano Berardi, Stefan Berghofe, Frederic Blanqui, Roel Bloo, Connor McBride, Sylvie Boldo, Ched Brown, Martin Bunder, Carsten Butz, Paul Callaghan, Venanzio Capretta, kaustuv Chaudhuri, Horatiu Cirstea, Catarina Coquand, Thierry Coquand, Judicael Courant, Pierre Courtieu, Anuj Dawar, Wil Dekker, Louise Denis, Mike Dewar, Gilles Dowek, Jean-Christophe Filliatre, Herman Geuvers, Juergen Giesl, Erich Graedel, Ferrucio Guidi, Chris Hankin, Therese Hardin, John Harrison, Hugo Herbelin, Daniel Hirschkoff, Martin Hoffmann, Randall Holmes, Jim Howe, Marieke Huisman, Paul Jackson, Tudor Jebelean, Assaf Kfoury, James McKinna, Frederic Lang, Marina Lenisa, John Longley, Marino Miculan, Eugenio Moggi, Cesar Munoz, Pavel Naumov, Masaki Nakamura, Rob Nederpelt, Catuscia Palamidessi, Larry Paulson, Simon Peyton-Jones, John Power, Christophe Raffalli, Silvio Ranise, Julian Richardson, Konrad Slind, Jan Smith, Ian Stark, Juergen Stuber, Laurent Thery, Simon Thompson, Franklyn Turbak, Andrzej Trybulec, Betti Venneri, Daria Walukiewicz, Burkhart Wolff, Wai Wong, Hongwei Xi.

Last but not least, I am grateful for Professor Deepak Kapur, editor-in-chief of the Journal of Automated Reasoning for his interest in a

special issue on the themes of the workshop to be included in the journal and for his valuable advice and feedback. Gail Pieper handled the manuscripts most efficiently and to her I am deeply thankful.

I will stop by quoting Robert Boyer of the University of Texas, famous for his Boyer-Moore theorem prover amongst other things:

"cheers for Automath's amazing influence on so many of us."

Fairouz Kamareddine

September 2002.

References

1. L.S. van Benthem Jutting. A Translation of Landau's "Grundlagen" in AUTOMATH. Technical report, Eindhoven University of Technology, 1976.
2. L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the Automath system.* PhD thesis, Eindhoven University of Technology, 1977. Published as Mathematical Centre Tracts nr. 83 (Amsterdam, Mathematisch Centrum, 1979).
3. N.G. de Bruijn. AUTOMATH, a language for mathematics. Technical Report 68-WSK-05, T.H.-Reports, Eindhoven University of Technology, 1968.
4. N.G. de Bruijn. The mathematical language AUTOMATH, its usage and some of its extensions. In M. Laudet, D. Lacombe, and M. Schuetzenberger, editors, *Symposium on Automatic Demonstration*, pages 29–61, IRIA, Versailles, 1968. Springer Verlag, Berlin, 1970. Lecture Notes in Mathematics **125**; also in [12], pages 73–100.
5. N.G. de Bruijn. Reflections on Automath. Eindhoven University of Technology, 1990. Also in [12], pages 201–228.
6. T. Coquand and G. Huet. The calculus of constructions. *Information and computation*, 76(2/3):95–120, February/March 1988.
7. H.B. Curry and R. Feys. *Combinatory Logic I.* Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1958.
8. N. G. de Bruijn. Lambda-Calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser Theorem. *Indag. Mat.*, 34(5):381–392, 1972.
9. W.A. Howard. The formulas-as-types notion of construction. In [13], pages 479–490, 1980.
10. T. Laan. *The Evolution of Type Theory in Logic and Mathematics.* PhD thesis, Eindhoven University of Technology, 1997.
11. E. Landau. *Grundlagen der Analysis.* , Leipzig, 1930.
12. Rob Nederpelt, J. H. Geuvers, and Roel C. de Vrijer. *Selected Papers on Automath.* North-Holland, Amsterdam, 1994.
13. J.P. Seldin and J.R. Hindley, editors. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press, New York, 1980.