Computational Tools for Mathematical Collaboration

Stephen M. Watt

26 September 2011

Domnule Rector Ioan Talpoș,
Domnilor Prorectori Stefan Balint, Florin Foltean și Pia Brânzeu,
Domnule Cancelar Dan Radu Moga,
Membri ai Senatului,
Domnule Decan Viorel Negru,
Colege și Colegi,
Doamnelor și Domnilor,

Accept cu modestie și deosibită considerație onoarea care mi se face astăzi. Legătura permanentă cu Universitatea de Vest din Timișoara, consemnată prin acest titlu adaugă un nivel simbolic colaborării de lungă durată cu colegii de la acestă universitate, pentru care am un respect profund.

Research collaboration has always been important and has recently become even more so. I wish therefore to address a topic I believe to require attention now, and that is software support for mathematical collaboration.

Over history, mathematics has been, by and large, a solitary pursuit. Like the classic author, the mathematician would work toward a goal, sometimes communicating with colleagues by letter or in person, but the work itself took place alone, and the results appeared as a publication with a single author. It is not surprising, therefore, that software tools for mathematics, such as Macsyma, Maple and Mathematica, have been designed with the conception of mathematical work as a solitary activity. Indeed, it has been observed [1] by Melvin Conway, the inventor of co-routines, that an organization produces software that replicates the communication structure of the organization itself. This is as true for mathematical software as for any other. Software tools for mathematics, including numerical programming environments, computer algebra systems and automated proof assistants, have been designed around the concept of mathematics as a solitary pursuit.

Relatively recently, collaboration, both tightly-knit and loosely-coupled, has become important for mathematics. This has been gradual, and is even now not

universally accepted as important by mathematicians. Nevertheless, we have meaningful examples, such as the systematization of the Bourbaki collective and the classification of finite simple groups. Within the past few years, the use of Internet web logs has allowed faster communication leading to a new level of collaboration. This is exemplified by the Polymath project [2], launched two years ago as the brainchild of Professor Timothy Gowers of Cambridge. His goal was to attempt to solve a problem through the collaboration of many individuals online. The trial problem was to find an elementary proof of a special case of the density Hales-Jewett theorem. Within weeks, Gowers announced that the Polymath participants had found such a proof, and that it could moreover be generalized to prove the full theorem. What, then, is the implication for mathematical software?

I would suggest that mathematical software must be reconsidered, in all its aspects, as a platform for collaboration. A new kind of mathematical software should be as different from the present generation of software as the social web is from E-mail. One can imagine a world of systematized facts and theories, proofs and conjectures, algorithms, heuristics and referee certificates under construction in a shared creative space, with different views for different starting points, different levels of rigour, etc. This is equally applicable to the world of pure mathematics as it is to applied problems, such as creating an engineering model for an aircraft. While they may not have been phrased as such, over the years, issues related to supporting collaboration in mathematical software have arisen at every turn, and now we are faced with these challenges again. I can relate a few examples from personal experience over the past three decades:

In the early 1980s, I had the good fortune to be involved at starting point of the Maple project, led by Keith Geddes and Gaston Gonnet. This was a personally formative time, as I was starting a PhD in Computer Science at the University of Waterloo, and I remember well the weekly meetings we would have in designing the Maple system. In particular, I remember the evening when we were first able to save and load libraries of Maple code, and the sense of how this would open the door to multiple contributors. While writing and using numerical libraries was common by then, computer algebra systems did not support user-level libraries very well. Not only did Maple allow these, most of the Maple system itself was written in the Maple language [3, 4]. This was a fundamental change at the time, forcing the Maple language implementation to be robust and efficient, and ultimately engaging a community as collaborators in developing packages.

At the end of 1984, I was presented the opportunity to join the Scratchpad group at IBM Research. This group, led by Richard Jenks, was designing a new system for computer algebra based on the structures of modern algebra [5]. The motivation for this design was to avoid code duplication by using mechanisms for abstraction. By specifying the mathematical properties of arguments using type categories, programs could be written in their most general mathematical setting and then used in particular situations. In contrast, other systems had

separate implementations for each different situation. Scratchpad made use of so-called "parametric polymorphism" more than a decade before it found its way in to main-stream programming languages. My role in this project was to complete the programming language and to lead the implementation of a compiler. This system and its programming language were ultimately released as Axiom [6] and Aldor [7, 8]. In the design of Aldor, one of the questions we addressed was how teams that were working independently could produce libraries that could be used together. In a rich environment this is not as simple as it seems. The main issue is how to allow individual, heavily-used modules to be extended and enhanced with new mathematical properties introduced by separate parties and to achieve highly efficient code without forcing rebuilding of the entire system. The use of type categories and *ex post facto* extensions of data types supported these goals. Type categories, and the use of a dependent type system, provided the requisite level of abstraction and allowed highly composible programs. *Ex post facto* extensions of mathematical types allowed independent extension of the central mathematical concepts [9]. For those who are interested in programming language implementation, the main idea of *ex post facto* extension is that adding mathematical properties to a type does not require changing the data representation, so multiple extensions can be applied without necessitating recompilation.

A next step in support for collaboration, in my personal experience, was OpenMath [10, 11]. This was a collective effort of several groups, including my own, initiated in the mid 1990s to develop a language to share mathematical data among applications. A principal strength, as well as a key weakness, of the OpenMath data language was its extensibility. Because it allowed extensions, little was required in its core and so the details of how any particular mathematical data would be represented required community agreement. While OpenMath did not, of itself, achieve widespread use, it did serve as a key tool in understanding the main issues in mathematical data exchange. It served as the basis of the European Union OpenMath and MONET projects, which explored how mathematical problems and algorithms could be characterized to allow publication and discovery of mathematical web services, and has been used in the European Project SCIEnce of which UVT is a member.

Shortly following the beginnings of OpenMath, in 1996 the World Wide Web Consortium (W3C) formed a group to address how mathematics should be represented on the web. While typesetting systems, such as TEX, were useful for expression mathematics in print, new technology was needed for pages that could be dynamically resized and for which a more semantic structure was desired. Recognizing that this could be an activity of far-reaching consequences, I participated in this group from its outset, helping to form the XML-based data language MathML [12]. This data language provided mechanisms to describe both the appearance and the meaning of a wide range of mathematical constructs. After creating the MathML standard, one of the main tasks of the working group was to foster adoption of MathML so that it could be used in the principal web browsers and document processors in addition to mathematical

software. MathML has now been through a number of revision cycles, and it has adopted OpenMath as its extension mechanism.

As the OpenMath and MathML efforts were underway, a third direction was emerging. This was the organization of a research community around the theme of "Mathematical Knowledge Management." This involved a diverse group of parties, interested in mathematical publishing, organization of theorem proving systems, computer algebra and web technology, and with the common desire to understand and usefully manipulate the information in collections of mathematical documents and software. Mathematical work is a thought process and a discourse that involves may sorts of information. In any given area of mathematics, there will be objects of discourse. These will be subjects of definitions, theorems, conjectures, algorithms, facts and observations. The area of Mathematical Knowledge Management addresses how these can be organized, related and manipulated, addressing questions that will be important for collaborative mathematical software.

Most recently, I have been investigating collaborative pen-based interfaces for mathematics. The InkChat software combines voice and digital ink in a multi-party conversation with a shared canvas. Pen input allows rapid discussion with natural entry of mathematical formulae, sketching and 2D-dimensional highlighting. At this stage the work is aimed at multiple human participants, with data formats that allow software packages to manipulate the digital ink as well. InkChat can be used by students and teachers in the same room or by practicing collaborators on opposite sides of the planet. Representing the writing in the vendor-neutral InkML [13] format allows participants on diverse platforms to access and analyze handwriting, constructing shared higher-level objects. What is written can be treated by different recognizers, replayed step-by-step and taken in new directions.

These are a few steps in the direction of fully collaborative computational mathematics. We still have a world where most mathematical software packages do not support collaboration in any serious way and, with a few notable exceptions, different software packages talk to each to other only weakly. Mathematical software systems must evolve if they are to be tools for working collaboratively. Just as GitHub, Source Forge and Google Docs provide new work-flow models to support the shared creation of software and documents, so should our mathematical software tools provide new ideas for collaboration. As a community, we have not thought deeply about this. There are a host of questions. For example: What are the best ways to manage the shared creative spaces for mathematical computation? How should the steps of a shared computation be themselves represented and manipulated? How can a collaborating group navigate the frontier of a computation? How can alternative directions, specializations or generalizations be best handled? How can different requirements of rigour, efficiency or constructibility be best accommodated? How should the objects of such mathematical collaboration link with the wider universe of mathematical knowledge? Note that these questions are independent

of whether the objects of discourse are polynomials, domains, proofs, programs or automotive models. We do not need to answer all these questions before be begin. As we gather experience, it will guide us.

Just as the modern web is transforming communication, opening a new economy, and reforming politics, the modern web can impact mathematical computation. People are the important actors in this process and their shared work is the important thing. From a higher-level perspective, conferences such as SYNASC are also tools for mathematical collaboration. While the time scales and granularities are different, the future of mathematical computing is perhaps more like an accelerated conference than it is like writing Fortran programs in isolation.

Vă mulțumesc pentru atenție. Fie ca viitorul să ne aducă cât mai multe noi rezultate și invenții comune.

Bibliografie

[1] M.E. Conway, *How do Committees Invent?*, Datamation 14 (5): 28–31, April 1968.

[2] `http://polymathprojects.org`

[3] B.W. Char, K.O. Geddes, W.M. Gentleman and G.H. Gonnet, *The design of Maple: A compact, portable, and powerful computer algebra system*, Proc. Eurocal '83, Springer-Verlag LNCS 162, 101–115.

[4] B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, M.B. Monagan and S.M. Watt, *On the Design and Performance of the Maple System*, Proc. 1984 Macsyma Users' Conference, General Electric Corporation, 199–219.

[5] R.D. Jenks and B.M. Trager, *A Language for Computational Algebra*, Proc. Symposium on Symbolic and Algebraic Computation (SYMSAC 81), ACM Press, 6–13.

[6] R.D. Jenks and R.S. Sutor, *Axiom: The Scientific Computation System* Springer Verlag, New York 1992.

[7] S.M. Watt, P.A. Broadbery, S.S. Dooley, P. Iglio, J.M. Steinbach and R.S. Sutor, *A First Report on the A♯ Compiler*, Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC 1994), ACM Press, 25–31.

[8] S.M. Watt, *Aldor*, in *Handbook of Computer Algebra*, J. Grabmeier, E. Kaltofen, V. Weispfenning (eds), Springer Verlag, Heidelberg 2003, 265–270.

[9] S.M. Watt, *Post Facto Type Extension for Mathematical Programming*, Proc. Domain-Specific Aspect Languages (SIGPLAN/SIGSOFT DSAL 2006), 26–31.

[10] `http://www.openmath.org` .

[11] S. Dalmas, M. Gaëtano and S.M. Watt, *An OpenMath v1.0 implementation*, Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC 1997), ACM Press, 241–248.

[12] D. Carlisle, P. Ion, R. Miner (eds), R. Ausbrooks, S. Buswell, D. Carlisle, G. Chavchanidze, S. Dalmas, S. Devitt, A. Diaz, S. Dooley, R. Hunter, P. Ion, M. Kohlhase, A. Lazrek, P. Libbrecht, B. Miller, R. Miner, C. Rowley, M. Sargent, B. Smith, N. Soiffer, R. Sutor, S. Watt, *Mathematical Markup Language (MathML) Version 3.0,* W3C Recommendation, October 2010, `http://www.w3.org/TR/MathML`.

[13] S.M. Watt, T. Underhill (eds), Y-M. Chee, K. Franke, M. Froumentin, S. Madhvanath, J-A. Magaña, G. Pakosz, G. Russel, M. Selvaraj, G. Seni, C. Tremblay, L. Yaeger, *Ink Markup Language (InkML)*, W3C Recommendation, September 2011, `http://www.w3.org/TR/InkML`.

Entry in the Universitatea de Vest din Timișoara log of *Doctor Honoris Causa* recipients

Timișoara, 26 September 2011

It is only through cooperation that scientific communities, or indeed society as a whole, can reach the farthest and achieve the most. May our future therefore be filled with shared discovery.

Stephen Watt