Systems related to the FMathL vision

by Arnold Neumaier, University of Vienna

April 29, 2010

There are already many automatic mathematical assistants that provide expert help in specialized domains. Known classes include computer algebra systems, automated deduction systems, modeling systems, matrix packages, numerical prototyping languages, decision trees for scientific computing software, etc.. Such existing tools already provide partial functionality of the kind to be created in the project but only tied to specific applications, or with a limited scope.

This document describes a number of current systems related to the FMathL vision, and some of their limitations when viewed in the light of this vision. The PI's website (`www.mat.univie.ac.at/~neum/FMathL.html`) contains a large selection of additional resources and references to existing related systems.

LATEX (`http://www.latex-project.org`) is today's de facto standard for the creation, communication and publication of scientific documents, widely used by mathematicians, scientists, and engineers. It is fairly user-friendly and produces documents of excellent quality; however, it only specifies the syntax and the quotation structure (references to equations, theorems, tables, figures, papers, footnotes, endnotes, etc.) of the documents, leaving the semantics inaccessible.

Markup languages. OpenMath (`http://www.openmath.org`) is an extensible language for representing the semantics of mathematical objects as a structured text. Its purpose is to facilitate the exchange of mathematical information between computer programs, databases, or worldwide web documents, and to enable its display in a browser. But there is no intrinsic display form for OpenMath objects. MathML (`http://www.w3.org/Math`) is a low-level specification for describing mathematics as a basis for machine to machine communication, with emphasis on web applications. MathML deals principally with the presentation of mathematical objects (so that MathML generated display looks like nice ordinary mathematics). It only has limited facilities for dealing with content; but it can embed OpenMath constructs. OMDoc (`http://www.omdoc.org/omdoc`) is a semantics-oriented representation format and ontology language for mathematical knowledge extending the markup of OpenMath and MathML to the document and theory level of mathematical documents. The voluminous representations

```
<OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="int"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR> <OMV name="x"/> </OMBVAR>
      <OMA>
        <OMS name="sin" cd="transc1"/>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
<matrix>
  <matrixrow>
    <cn> 0 </cn> <cn> 1 </cn> <cn> 0 </cn>
  </matrixrow>
  <matrixrow>
    <cn> 0 </cn> <cn> 0 </cn> <cn> 1 </cn>
  </matrixrow>
  <matrixrow>
    <cn> 1 </cn> <cn> 0 </cn> <cn> 0 </cn>
  </matrixrow>
</matrix>
</math>
```

of $\int \sin x\, dx$ in OpenMath and of the matrix $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$ in MathML, taken from the above web sites, show that a markup format is not suitable as a modeling tool.

Mizar (`http://mizar.org`) is a formal language for specifying the syntax and semantics for mathematical texts in a way that allows them to be verified automatically by computer starting from set-theoretical axioms. A 16-year old journal entitled "Formalized Mathematics" specializes in publishing Mizar-generated mathematics. The Mizar input is difficult to read and create, but the output is reasonably readable, embedding the formulas in English (though often somewhat uneven) sentences. There are no capabilities to specify data-driven applications.

Common mathematical language. Making ordinary mathematical text completely comprehensible for the computer is an exceedingly difficult task, at present virtually impossible without much interactive assistence. Fully automatic partial work is reported in two Ph.D. theses by SIMON [14] and ZINN [18], which exhibit the many problems encountered in their attempts to turn some elementary number theory texts into checkable proofs. Therefore, the informal mathematical language must be somewhat restricted to be automatically analyzable. KAMAREDDINE [7] emphasizes the problems to be overcome for a full formalization acceptable to the typical mathematician, and introduces **Math-Lang** (KAMAREDDINE et al. [9, 10], VAN TILBURG [15]), a non-fully-formalized language close to the common mathematical language, for interactively annotating mathematics written in English and translating it into Mizar [8]. Currently, the system seems to be in an explorative stage of development and does not seem to have a structure that would lend itself easily to the specification of large optimization models or other applications.

A very interesting recent thesis by GANESALINGAM, *The Language of Mathematics*, gives a vivid account of the linguistic problems involved in the analysis of mathematical texts, and in particular offers solutions for the problem of unambiguously resolving ambiguities in the interpretation of mathematical formulas. Currently, there is no associated software available.

Naproche (`http://www.math.uni-bonn.de/people/naproche`) is an ongoing project that aims to translate natural mathematical text into a sequence of first order logic sentences. The input for the system is a plain LaTeX file, without any further annotation. However, the system only accepts a very small and rigid controlled natural language. The people from the project are working on making this language more flexible, but until now it is by far not flexible and rich enough to be of actual use for a mathematician. The system produces also output in the FOF-format of the TPTP (`http://www.cs.miami.edu/~tptp/`), the standard challenge database for current theorem provers; hence, by using the tools of the TPTP, a wide variety of theorem provers can be accessed.

The Z notation. The Z notation (`http://vl.zuser.org`) is a specification language based on Zermelo-Fraenkel set theory and first order predicate logic, available since 2002 as an ISO standard [5]. It is a mathematical specification notation, used by industry as part of the software and hardware development process for the highly reliable definition of algorithmic components, particularly for safety-critical systems. Z is intended to increase human understandability of the specified system and to allow the possibility of formal reasoning and development. Professional quality typeset documents based on Z specifications can be produced by a LaTeX interface provided by the tool set CADiZ (`http://www-users.cs.york.ac.uk/~ian/cadiz/`); this system also provides support for finding proofs related to Z specifications. Various other tools are available, including an interface to the theorem proving environment HOL. However, there is no support for nonalgorithmic mathematics such as abstract infinite sets or integrals. Moreover, the specifications themselves look more like programs than like mathematics.

Modeling systems. For the application to numerical problems, in particular large-scale optimization, a number of versatile modeling languages are available; see KALLRATH [6]. Widely used examples include AMPL (`http://www.ampl.com`) and GAMS (`http://www.gams.com`). They are very efficient for problems from the applications, with interfaces to all major state of the art optimization packages, providing the derivatives needed by these solvers automatically. But the current modeling languages also have serious limitations; in particular, they are not able to understand arbitrary mathematical formulae.

CVX is a software system for disciplined convex programming, allowing the user to specify nonlinear optimization problems with automatic verification of their convexity properties, in an easy to use syntax. While very specialized, it is an example of a system that checks the assumptions of a numerical method before trying it, and by its syntax enforces a healthy disciplined approach to model formulation.

Decision trees for software. The Decision Tree for Optimization Software by MITTELMANN (`http://plato.la.asu.edu/guide.html`) gives online advice about which software to use for which kind of optimization problem. Similar decision trees are discussed in the literature for a variety of problems in scientific computing; see, e.g., ADDISON et al. [1], RAMAKRISHNAN & RIBBENS [12], RAMAKRISHNAN et al. [13], HOUSTIS et al. [4], MAARUSTER et al. [11], BUNUS [2]. Although not a modeling tool, such decision trees have a natural place in a modeling system that automatically selects the solver to use.

Monet (`http://monet.nag.co.uk/monet`) was a project undertaken by an international consortium of academic institutions and industrial partners between 2002–2004. Its objective was to develop a framework for the description and provision of web-based mathematical services. Monet differentiated between *static descriptions* of mathematical objects based on XML-based technologies such as OpenMath and MathML and *dynamic processing* of mathematical objects based on Description Languages such as WSDL. Monet produced interesting work, which can potentially be reused in our system, but its emphasis on intricate web technologies neither widely known nor very popular might have squandered its chances of practical success.

Grammatical Framework. The Grammatical Framework (GF) (`http://www.grammaticalframework.org`) is a special-purpose programming system whose core is a generic grammar processor capable of recognizing and generating important fragments of many natural languages. It particularly addresses multilinguality, semantic conditions of well-formedness, language modularity and the reuse of grammars in different formats and as software components. It provides enough functionality for the restricted natural language support required for a formal language close to the traditional informal mathematical usage. Other language-related references of potential interest for the project are CARLSON et al. [3] for multilingual mathematics and Greenstone (`http://www.greenstone.org`) for multilingual software for building and distributing digital library collections.

WebALT (`http://webalt.math.helsinki.fi`) was a project funded by the eContent EU Programme between 2005–2007. It built upon the Grammatical Framework GF in order to produce language-independent mathematical didactical material, targetting primarily students of undergraduate mathematics. It produced as its core deliverable a mathematical grammar for the language-independent encoding of exercises in mathematics, as well as mathematical grammars for a number of European languages to enable the automatic natural language generation of mathematical texts into these languages.

Wolfram—Alpha Computational Knowledge Engine (`http://www.wolframalpha.com/`) has recently (May 2009) launched to great expectation. It defines itself as a system for knowledge-based

computing aiming to make "all systematic knowledge immediately computable by anyone". It is built on top of Wolfram Research's Mathematica system, although at present it does not accept general Mathematica input. Instead it accepts queries in natural language with formulas. The Wolfram—Alpha Computational Engine summons impressive computational capabilities; by contrast, its capability to understand and communicate with the user appears rather impoverished. Reasonable input in natural language is often not understood. Furthermore, the system is currently rather sensitive to changes in the syntax of the queries, which seems to be indicative of a lack of any real semantic comprehension. $z^2 = x^2 + y^2$ give very different results, and neither result mentions Pythagora's Theorem.

Computer algebra systems and numerical prototyping languages. Mathematical software packages such as e.g. Mathematica (`http://www.wolfram.com/products/mathematica`), MuPAD (`http://www.mupad.de`) and Maple (`http://www.maplesoft.com`) are computer algebra systems with lots of built-in functionality. The systems Matlab (`http://www.mathworks.com/products/matlab`), Scilab (`http://www.scilab.org`) are programming environments for efficient numerical computation and prototyping, easy to use for the working mathematician. All these are widely used but do not give the user access to the mathematical meaning of the constructs, so that one cannot use the defined relations for anything but for computations.

Logical frameworks. There are a large number of logical frameworks and theorem proving systems which allow the verification of (carefully specified) proofs for mathematical statements, primarily of properties of programming languages and logics. Widely used systems include Coq (`http://pauillac.inria.fr/coq`), HOL light (`http://www.cl.cam.ac.uk/~jrh13/hol-light`), PVS (`http://pvs.cl.sri.com`), and Twelf (`http://www.cs.cmu.edu/~twelf`). Many of these systems can be used to prove nontrivial mathematics; see, e.g., WIEDIJK [16] (100 formally proved theorems, ranging from the fundamental theorem of algebra to Brouwer's fixed point theorem), and WIEDIJK [17] (a comparison of provers for the irrationality of $\sqrt{2}$). But proofs are generally unreadable manually (except by expert users). Typically, statements and proofs must be provided in a program-like structure far removed from mathematical practice. This makes these systems unsuitable for mathematical modeling applications.

The **Theorema** package (`http://www.risc.uni-linz.ac.at/research/theorema`) is a Mathematica-based prototype extended by facilities for supporting logically rigorous proving of mathematical statements. It aims to assist the working mathematician through all the phases of the mathematical problem solving cycle (specification, exploration, proving, programming, and writing up). **Theorema** allows users to interactively build proofs for mathematical theorems, with nested-cell proofs that are readable and can be expanded or shortened in a user-controlled manner. Although input and output appear in a form acceptable for mathematicians, the results are often somewhat stilted. More importantly, there are no capabilities for specifying data-driven applications.

There is a small-scale project that aims at formalizing elementary analysis rigorously at the level of a high-school student. The argument used to support the FEAR project (`http://www.nwo.nl/projecten.nsf/pages/2200126954`), are in line with our own goals: *"The main weakness of current proof assistants is that mathematically they are not powerful enough. They are very good at keeping track of all of the details of a mathematical proof: one routinely checks proofs using these systems that have hundreds, or even thousands, of cases. However, they are currently not very good at taking by themselves steps that a human mathematician considers to be trivial. One of the most important class of steps that are not supported well are problems that a high school student can solve without difficulty. We call these high school problems."* (`http://www.cs.ru.nl/~freek/notes/oc2004.pdf`)

References

[1] C.A. Addison, W.H. Enright, P.W. Gaffney, I. Gladwell, and P.M. Hanson. Algorithm 687: a decision tree for the numerical solution of initial value ordinary differential equations. *ACM Trans. Math. Software*, 17:1–10, 1991.

[2] P. Bunus. A simulation and decision framework for selection of numerical solvers in scientific computing. In *Proc. 39th ann. Symp. Simulation, IEEE Computer Society, Washington*, pages 178–187, 2006.

[3] L. Carlson, J. Saludes, and A. Strotmann. State of the art in multilingual and multicultural creation of digital mathematical content, 2005. Web Advanced Learning Technologies, Manuscript.

[4] E.N. Houstis, A.C. Catlin, N. Dhanjani, and J.R. Rice. MyPYTHIA: a recommendation portal for scientific software and services. *Concurrency Computat.: Pract. Exper.*, 14:1481–1505, 2002.

[5] ISO. International standard ISO/IEC 13568:2002 information technology – Z formal specification notation – syntax, type system and semantics, 2002.

[6] J. Kallrath. *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis*. Vieweg, 2002.

[7] F. Kamareddine. Motivations for MathLang, 2005. Slides.

[8] F. Kamareddine, M. Maarek, K. Retel, and J.B. Wells. Gradual computerisation/formalisation of mathematical texts into Mizar, 2004. Manuscript.

[9] F. Kamareddine, M. Maarek, and J.B. Wells. *Flexible encoding of mathematics on the computer*, pages 160–174. Springer, Berlin, 2004.

[10] F. Kamareddine, M. Maarek, and J.B. Wells. MathLang: An experience driven language of mathematics. *Electronic Notes in Theoretical Computer Science*, 93C:123–145, 2004.

[11] S. Maaruster, V. Negru, D. Petcu, and C. Sandru. Intelligent front-end for solving nonlinear systems of differential and algebraic equations. *J. Math. Sci.*, 108:1139–1151, 2002.

[12] N. Ramakrishnan and C.J. Ribbens. Mining and visualizing recommendation spaces for elliptic pdes with continuous attributes. *ACM Trans. Math. Software*, 26:254–273, 2000.

[13] N. Ramakrishnan, J.R. Rice, and E.N. Houstis. GAUSS: an online algorithm selection system for numerical quadrature. *Adv. Engin. Software*, 33:27–36, 2002.

[14] D.L. Simon. *Checking Number Theory Proofs in Natural Language*. PhD thesis, Univ. of Texas, Austin, 1990.

[15] P. van Tilburg. Exploring the core of MathLang, 2006. Manuscript.

[16] F. Wiedijk. Formalizing 100 theorems, 2007. WWW-Document.

[17] F. Wiedijk. The seventeen provers of the world, 2007. WWW-Document.

[18] C. Zinn. *Understanding Informal Mathematical Discourse*. PhD thesis, Univ. Erlangen, 2004.