

Deliverable 3.5:
User-oriented tools

Mathematical Knowledge Management Network
MKMnet, IST-2001-37057

Summary

The present document is the result of **Task 3.5: *User-Oriented Tools***, of the **Work Package 3: Varied Presentations of Mathematics and Varied User Needs**.

The principal objective of this task is to make an inventory of the users needs and of the different solutions provided by existing tools.

The sections have been compiled by various partners in the project, using information from other groups when necessary.

The complete report has been collected and integrated by Fairouz Kamareddine, Heriot-Watt University, Edinburgh.

1 Current state of User-Oriented Tools

There are concerns about providing appropriate navigation through mathematical knowledge bases and providing access at the appropriate level of detail including both informal overviews and formal proofs.

Where learning is concerned, there are issues about providing appropriate examples, helping to correct misunderstandings and promoting further exploration. All of these have implications not just on the appearance of the user interface but also on what is actually represented in a knowledge base and how it is inter-related.

Technologically, much can be achieved in this direction with MathML, a W3C standard aimed to include presentation primitives for mathematical notation in \LaTeX format into XML. MathML is already (partially) supported by the most recent versions of the commercial browsers Internet Explorer and Netscape via special plug-ins, and natively by the open source browsers Mozilla and Amaya. The GTK widget GtkMathView (developed at the Department of Computer Science of Bologna, member of this Consortium) also provides a complete and efficient viewer for MathML documents along with a conversion utility from MathML to PostScript.

The problem of recovering presentation from content may be conveniently addressed by means of notational stylesheets written in XSLT (the standard mean of the World Wide Web Consortium for transforming and rendering XML documents). The OpenMath project is doing some work here as regards the rendering of OpenMath into MathML, rendering OMDoc documents, including texts and OpenMath objects is now also performed.

To achieve this user-oriented approach to mathematical knowledge management, those developing tools for knowledge management will need to understand the needs of users. This can come from dialogue between the members of the Network as they have a wide range of backgrounds and interests in mathematical knowledge management. Also, specific research can be done with users, for instance, research mathematicians, to investigate their tasks and their need for computer-based tools. User-oriented analysis of current systems for mathematical knowledge management, for instance, automated theorem provers, could suggest projects for closing the gap between users and such systems. Analysis of existing mathematical journals could also suggest the sorts of support that mathematical authors and editors need.

Various sites in the network have worked on user-oriented tools. This research is summarised in the following sections.

1.1 User interface for interactive theorem provers

In [2], it is argued that interactive theorem proving systems for mathematics require user interfaces which can present proof states in a human understandable way. Often the underlying calculi of interactive theorem proving systems are problematic for comprehensible presentations since they are not optimally suited for practical, human oriented reasoning in mathematical domains. The recently developed core theorem proving framework [1] is an improvement of traditional calculi and facilitates flexible reasoning at the assertion level. The paper makes use of core’s reasoning power and develops a communication layer on top of it, called the *task layer*. For this layer the authors define a set of manipulation rules that are implemented via core’s calculus rules. They thereby obtain a human oriented interaction layer that improves and combines ideas underlying the window inference technique [14], the proof by pointing approach [3], and the focus windows of [13].

1.2 User-friendliness and semantic authoring

In [11, 12], a GTK widget for rendering MathML and interacting with MathML markup called GtkMathView, is described. As such it is not correct to consider GtkMathView as a real tool, it is rather a library that a developer can use to build an application. In fact the main point of GtkMathView is that it is *open-ended* and the developer can build on it the interaction capabilities desired for the application.

Recently, the Bologna team has approached the problem of semantic authoring mathematical documents (either formulas or proofs). By semantic authoring, it is meant, a process where the outcome is a document that is machine understandable at a content-oriented level, not just purely presentational. At the same time, the Bologna team has tried to provide a tool that is user-friendly, by this meaning that the author works in a comfortable “high-level” environment where support is provided for the presentation of rich mathematical notation and the interaction with the authored document. GtkMathView provides adequate support for the presentational part of the tool. As for the actual authoring part, we realized that the editing capabilities available in several similar tools are not satisfactory for two main reasons:

- editing is slowed down by an intensive use of the pointing device (the mouse) for accessing palettes of symbols and menus for the insertion of mathematical constructs.
- editing typically occurs at a simple presentational level with no or little

interest in the semantics underlying the presentation

By combining the capabilities of a WYSIWYG editor and a \TeX -to-MathML translator, the team has developed a MathML editor based on \TeX syntax¹ where the author types \TeX markup as he or she would do in a normal text editor. The markup is parsed as typing occurs, an internal structured model of the mathematical expression is built which is subsequently processed by means of XSLT stylesheet producing a corresponding MathML document that GtkMathView can display. The advantages of this approach can be summarized as follows:

- the editing environment is comfortable as the user is effective in typing the expressions and at the same time the expressions are presented in a graphical way;
- the user is free to define domain-specific macros in order to edit the expressions at a more content-oriented level. The use of XSLT for the rendering of the structured model allows the user to extend the editor to support new notations and/or new macros.

Remarkably not all the issues regarding semantic authoring can be solved effectively by the definition of new macros. In fact it is not reasonable to imagine authors typing something like $\backslash\text{plus}\{x\}\{y\}$ in place of $x+y$ for each occurrence of the ubiquitous plus operator. Thus it may be the case that the authored document needs further refinement in order to acquire non-ambiguous meaning. In a context where mathematics is formalized (think of proofs in the library of the Coq proof assistant), it is possible to combine editing and searching for disambiguating symbols occurring in expressions. A first version of such tool has been implemented in a prototype proof-assistant based on the Calculus of Coinductive Constructions.

1.3 For Learners and Teachers: ActiveMath

The ActiveMath learning environment "jEditOQMath"² (developed at the DFKI and the University of Saarland) is a web-server offering a learning experience featuring individualized presentation, access to mathematical systems, intelligent advise.

ActiveMath content is OMDoc with mathematical objects in OpenMath. The usage of OMDoc within ActiveMath is described in more details in [7].

¹See <http://helm.cs.unibo.it/software/editex/>

²See <http://www.activemath.org/>.

Instead of encoding its content in presentation languages like most learning environments (for example in LaTeX or MathML-presentation) ActiveMath requires its authors to input OMDoc and offers, in exchange, the delivery of the content in four formats (HTML, PDF, SVG, XHTML+MathML) [6]. This process of presentation provides more than visual formulae: all mathematical symbols in the formulae have a precise meaning which can be introspected by the learner. For example, the learner can click on each symbol to see its common-name and browse any available definition. Presumably, a little *formula explorer* should be easily implementable. For a limited amount of expression, this markup even allows the learner to copy and paste subparts of the formulae from the presented content to the interactive exercise sessions with mathematical systems.

To make the knowledge accessible in a user-adapted way and in an educational way, more and particular annotations (with metadata) need to be added. These are of course different from metadata you need for other applications. ActiveMath also provides facilities of an intelligent user-environment: using metadata annotations at the level of the OMDoc *items* (more or less equivalent to the paragraph level), a course-generation facility allows the learner to build a book with content planned for his needs and particularities (e.g. for its education field or level).

Interactive exercises in ActiveMath are thus far: Multiple-Choice-Questions Mapping exercises and Fill-In-the-Blanks interactive exercises supported by a CASS. The usage of mathematical systems currently assumes a simple console for the interaction-cycle:

input/compute/evaluate/feedback.

The language of the CAS is used and the display is in *ASCII-art*. Experiments for displaying MathML are ongoing.

As indicated, the content in ActiveMath has to be encoded in OMDoc. We review the current authoring methods in the next section.

1.4 Tools for Input of Mathematical Documents

Existing authoring tools for writing mathematical documents are mostly not applicable to produce OMDoc documents as required by ActiveMath. There exist several tools for producing textual content and items annotated with metadata.

The delicate part in authoring OMDoc content is the input of mathematical objects. We are aware of the following tools for this purpose:

- The mathematical systems that can produce OpenMath could be usable.

- The QMath processor [4] has an input syntax very similar to that of mathematical systems and accepts unicode characters.
- An OpenMath graphical editor would do. As of this date, Jome is the only one known (see <http://mainline.essi.fr>).
- Converters from other languages with a partial guess for the conversion, e.g., LaTeX2OMDoc.

The OpenMath objects (just as the MathML-content objects), need to be appropriately picked, formulated, and combined. Finding the appropriate symbol is the first challenge to be solved by an author. Browsing OpenMath content-dictionaries is a first approach to do so. Often however, and this is especially the case in OMDoc, a whole theory needs to be taken in consideration (be *imported*) for a symbol to be used. Automated support is desirable and possible.

New symbols can be defined. This requires that at least more information is provided to the presentation processes in order to present the new symbol as part of mathematical formulae.

ActiveMath provides an environment that allows the authors to develop their content, putting them into OMDoc as source XML with QMath-encoded formulae, and test/look at their content within the learning environment. The round-trip write/process/test/correct is of importance as authoring content level encoded formulae is very new to most authors.

1.5 The FoC tool

The FoC language allows a programmer to express mathematical knowledge using concepts which are close to mathematical practice. The FoC language provides to a programmer notions of species, collections, inheritance, signatures, properties, definitions, theorems (see [15]) which reflect usual mathematical concepts. These concepts should be available to other people not necessarily knowing the FoC language.

To achieve this goal, a first task has been to design a translation from FoC sources into OMDoc (see [5]) documents (see [8]). This enabled the FoC project to have a first contact with XML based standards used inside MKM. The purpose of the translator was to combine tools inside the FoC compiler and general XML tools that would finalize the translation. While developing the translator it became more adequate to design a *dedicated* XML format for FoC where the internal structure and the coherence of FoC sources could be maintained. This lead to the FoCDoc format (see [9]) which is an XML encoding of FoC structures.

The main advantage of the FoCDoc format is that it is independent of a particular way some feature is encoded. For instance it was difficult to obtain a translation from an OMDoc document into a FoC program. This was also the case when the OMDoc document was the translation of a FoC program! On the contrary FoCDoc was designed to reflect FoC programs and further processing could be done on FoCDoc documents.

It was thus important to achieve further treatments on the FoCDoc format. In order to avoid difficulties already encountered when dealing directly with OMDoc or plain OpenMath we decided to use MathML as a target language for a documentation system (see [10]). An important advantage of MathML over more powerful but still emerging standards is that it is a well established and widely supported language for viewing mathematical documents.

Of course we did not redesign routines inside the FoC compiler but relied on the FoCDoc format which is a faithful view of FoC sources. Following other MKM nodes we decided to use Stylesheet translations to produce our MathML documentation. Since FoC sources reflect the semantics of mathematical structures such as rings or fields, we decided to produce only MathML-Content documents and to rely on D. Carlisle's stylesheet translator to obtain MathML-Presentation documents which are viewable using almost any MathML enabled browser.

One important problem we had while writing our translator was that MathML was not able to express the complex notions which are dealt with in the FoC library. We thus concentrated on the different extension mechanisms available in MathML and we used them to achieve a complete MathML documentation for the FoC library.

As a result of this MKM funded work we now have a tool which is able to produce documentation for the FoC library which is readable by any mathematician using any MathML enabled browser.

Inside this part of the project we are also beginning to consider further exchanges between FoC and other tools. One such exchange could be to provide FoC's computing capabilities to other theorem provers or to use proofs already made inside other theorem provers inside a FoC program.

These features require bidirectional communications between FoC and another tools using for instance mathematical buses (see [17, 16]). We are considering to use the full power of OpenMath to achieve this goal.

2 Interoperability among the user-oriented tools

Semantic authoring continues to face major problems:

- Output: How to output mathematical texts?
- Retrieval: How to find the right item?
- Input and editing: How to edit and input items and texts?

There is a clear need for pervasive search facilities, and for user-oriented editing and input/output tools. These issues have been some of the major ones discussed at the User Interface and Mizar worksops in September during the MKM conference. Each of the partners in the consortium have faced these problems and have proposed solutions, and each of us know that much more work is needed to be able to input and output our mathematical knowledge into and out of the computer. Mizar has shown very good success at dealing with the issue of data mining and this research is leading the way.

Interoperability among the user-oriented tools is even more difficult. Collaborations between the Bologna system HELM and the DFKI system ActiveMath have led to experimentations of discrepancies amongst the "semantic levels" and this in turn implied major interoperability issues. From this, it was concluded that "content math", (as in the level of OpenMath or computer-algebra-systems), is one step fuzzier than semantic-math (which, in principle, has a fully defined semantic, including its logical foundation).

In summary, even within one system, there is still much work to be done on input, output, editing and retrievals to make the system as user-friendly as possible. When it comes to bridging different systems the difficulties multiply. Some partners in the net believe that interoperability seems to be only guaranteed by the usage of web-services, but this is a topic which hasn't had the time to mature in order to get progress into more user-friendly tools.

References

- [1] Serge Autexier. *Hierarchical Contextual Reasoning*. PhD thesis, University of the Saarland, 2004. to appear.
- [2] Malte Hübner and Christoph Benz Müller and Serge Autexier and Andreas Meier. Interactive Proof Construction at the Task Level. In *Proceedings of the Workshop User Interfaces for Theorem Provers (UITP 2003)*, pages 81-100, Rome, Italy, MMIII ARACNE EDITRICE S.R.L. (ISBN 88-7999-545-6). 2003.
- [3] Yves Bertot, Gilles Kahn, and Laurent Thery. Proof by pointing. In *Theoretical Aspects of Computer Science (TACS)*, 1994.

- [4] A. González Palomo. QMATH: An authoring tool for omdoc, Jan 2002. See <http://www.matracas.org/> for updated info.
- [5] Michael Kohlhase. *OMDOC: An Open Markup Format for Mathematical Documents (Version 1.1)*. <http://www.mathweb.org/omdoc>, 2003.
- [6] P. Libbrecht, C. Ullrich, and S. Winterstein. An efficient presentation-architecture for personalized content. In R. Tolksdorf and R. Eckstein, editors, *Proceedings of Berliner XML Tage 2003*, pages 379–388, 2003.
- [7] E. Melis, J. B. E. Andres, A. Frischauf, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. Knowledge representation and management in ACTIVE MATH. *Annals of Mathematics and Artificial Intelligence, Special Issue on Management of Mathematical Knowledge*, 38(1-3):47 – 64, 2003. Volume is accessible from <http://monet.nag.co.uk/mkm/amai/index.html>.
- [8] Manuel Maarek. Conception d’une librairie OMDoc pour FoC. Rapport de stage de D.E.A, Université Paris 6, September 2002. In French.
- [9] Manuel Maarek and Virgile Prévosto. Focdoc: The documentation of foc. In *Calcuemus 2003 Proceedings*, September 2003.
- [10] Yvan Noyer. Conception et réalisation d’un traducteur XSLT de FoC-Doc vers MathML. Rapport de stage de D.E.S.S, Université Paris 6, September 2003. In French.
- [11] L. Padovani, “MathML Formatting”, Ph.D. Thesis, Technical Report UBLCS-2003-03, Dept. Computer Science, Bologna, Italy, 2003.
- [12] L. Padovani, “MathML Formatting with \TeX Rules and \TeX Fonts”, in TUGBoat, The Communications of the \TeX Users Group, Volume 24, to appear.
- [13] Florina Piroi and Bruno Buchberger. Focus windows: A new technique for proof presentation. In Jaques Calmet, Belaid Benhamou, Olga Caprotti, Laurent Henocque, and Volker Sorge, editors, *Artificial Intelligence, Automated Reasoning and Symbolic Computation*, number 2385 in LNAI, pages 337–341. Springer, 2002.
- [14] Peter J. Robinson and John Staples. Formalizing a hierarchical structure of practical mathematical reasoning. *Journal of Logic Computation*, 3(1):47–61, 1993.

- [15] FoC Project. *The FoC System Reference Manual Version 0*. SPI LIP6, 2003.
- [16] J. Zimmer and L. Dennis. Inductive Theorem Proving and Computer Algebra in the MathWeb Software Bus. In *Proceedings of the 10th CALCULEMUS Symposium 2002*, Marseille (France), July 2002.
- [17] Jürgen Zimmer and Michael Kohlhase. System description: The MathWeb software bus for distributed mathematical reasoning. In *Proceedings of the 18th International Conference on Automated Deduction*, LNAI, 2002.