An Environment for Building Mathematical Knowledge Libraries

Florina Piroi[*] Bruno Buchberger

Research Institute For Symbolic Computation,
4232 Hagenberg, Austria

{florina.piroi, bruno.buchberger}@risc.uni-linz.ac.at

June 5, 2004

Abstract

In this paper we identify the organizational problems of Mathematical Knowledge Management and describe tools that address one of these problems, namely, the additional annotation of formalized knowledge. We describe, then, how the tools are realized in the frame of the *Theorema* system.

1 Introduction

The aim of the new research area "Mathematical Knowledge Management" (MKM) is the computer-support (partial or full automation) of all phases of the exploration of mathematical theories:

- invention of mathematical concepts,
- invention and verification (proof) of mathematical propositions,
- invention of problems,
- invention and verification (correctness proofs) of algorithms that solve problems,

and the structured storage of concepts, propositions, problems, and algorithms in such a way that, later, they can be easily accessed, used and applied.

MKM in this broad sense is an essentially logical activity: All formulae (axioms and definitions for concepts, propositions, problems, and algorithms) must be available in the coherent frame of a logical system, e.g. some version of predicate logic and the main operation of MKM on these formulae is essentially formal reasoning (in particular formal proving).

The *Theorema* system is one of the systems whose emphasis is on this logic aspect of MKM, which we think is the fundamental aspect of future MKM. Some papers on

[*]partially supported by the RISC PhD scholarship program of the government of Upper Austria and by the FWF (Austrian Science Foundation) SFB project P1302

the logical aspects of MKM within *Theorema* are [12, 5]. The question of computer-supported invention of mathematical knowledge within *Theorema* is treated in [9], the question of computer-supported algorithm synthesis within *Theorema* is treated in [6, 8] and [10].

On the surface of MKM, however, we are faced also with many additional organizational problems, which are important for the practical success of MKM:

a. The translation of the vast amount of mathematical knowledge which is available only in printed form (in textbooks, journals etc.) and which has to be brought into a form (e.g. LaTeX), in which it can be processed by computer algorithms. This is the problem of "digitization" of mathematical knowledge, see e.g. [25] for a survey on the existing projects in this area. The *Theorema* project is not engaged in this area of MKM.

b. The translation of digitized mathematical knowledge, for example in the form of LaTeX files, into the form of formulae within some logical system, e.g. predicate logic so that, afterwards, they can be processed by reasoning algorithms (in particular automated theorem provers). Many current projects are addressing this question, see e.g. MathML [26], OpenMath [13]. The *Theorema* project is not engaged in this area of MKM.

c. The organization of big collections of formulae, which are already completely formalized within a logic system (e.g. predicate logic) in "hierarchies of theories". At the moment, the largest such collection is Mizar [21]. Among other existing ones we mention MBase [19], the Formal Digital Library project [1], the NIST Digital Library of Mathematical Functions [20], the libraries of the theorem provers Isabelle [17], PVS [23], IMPS [16], Coq [14].

   The subproblem c., again, has two sub-aspects:

   c1. The organization of formalized mathematical knowledge by means of mathematical / logical structuring mechanisms like domains, functors, and categories. (Within *Theorema*, these questions are treated, for example, in [7].)

   c2. The additional annotation of formalized mathematical knowledge by "labels", so that blocks of mathematical knowledge can be identified and combined in various ways without actually going into the "semantics" of the formulae.

For the above and other overall views of MKM see [11, 2] and [4].

This paper exclusively deals with the subproblem c2. Traditionally, mathematical texts (collections of formulae) are organized in chapters, sections, subsections, etc. and individual formulae may have additional descriptive key words like "Definition", "Theorem", "Lemma" etc. and subformulae may also have individual labels like "(1)", "(2)", "(a)", "(b)", or "(associativity)" etc. All these external descriptors of formulae are used as (hierarchical) labels, which have no actual logical meaning or functionality, but they are only used for quick (and hopefully unique) referencing of formulae in big mathematical texts. Also, parts of large mathematical texts may be available in various files and often we will like to include text from various files as parts of another.

In traditional mathematical texts, these various descriptors of blocks of formulae and individual formulae are usually assigned in an ad hoc way. However, for the future computer-based management of mathematical knowledge, tools for generating and using these descriptors for accessing pieces of mathematical text and individual formulae are of vital practical importance.

In this paper, we report on tools which we developed recently for supporting the automated generation of unique labels (descriptors) for formulae and collections of formulae within the *Theorema* system and for using these labels in a systematic way for the build-up of coherent formal mathematical texts, i.e. collections of formulae within the *Theorema* version of predicate logic. Although these tools have been developed for *Theorema*, the design principles of the tools are independent of *Theorema* and may be useful also for other systems of formal mathematics. The design of the tools is based on ideas of the second author, the concretization for *Theorema* and actual implementation is part of the first author's forthcoming PhD thesis [24].

The plan of the paper is as follows: In section 2 we review the work that is going on in the area of Mathematical Knowledge Management and we give the main design idea of our tools, as a mathematical document editing environment. In section 3 we will describe how they are integrated in *Theorema*. We will end with conclusions and remarks on future work in section 4.

2 Towards Mathematical Document Parsing

When thinking of a mathematical knowledge base, most of us will, more or less, have in mind a big collection of formulae (definitions, theorems, etc.) organized in some hierarchical structure. Usually, this knowledge is to be found in specialized books, which have the big disadvantage of presenting the information in a static way. Searching in them can only be done syntactically and is time consuming. An important step forward was done by using computers to electronically store and search within mathematical documents (organizational problem a. in the previous section).

As the Internet became one of the most handy and used tools for finding information, it was a natural step to employ it for making mathematical knowledge widely available. Still, for some time, mathematical formulae were displayed only as graphics.

Using the MathML recommendation of W3C [26], it is now possible to display and communicate formulae. Being an application of XML, MathML benefits from the existing tools that manipulate XML files. Though it does offer some semantics of the symbols in the mathematical formulae, the set of these symbols is too restricted when compared to those used by working mathematicians. To ameliorate this situation projects like OpenMath [13] and OMDoc [18] emerged. The OpenMath standard concentrates on representing mathematical objects together with their semantic meaning, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. Though it is not exactly so, one can view OpenMath as extending the MathML capabilities by using "content dictionaries" where mathematical symbols are defined syntactically and semantically. OMDoc is an extension of OpenMath and MathML, adding capabilities of describing the mathematical context of the used OpenMath objects.

An important drawback of the standards mentioned above is that the coherence of the different documents (e.g. content dictionaries) is not automatically checked. This has to be done by a human, the task being rather difficult because the representation formats are not human oriented. This representation confronts us with another issue, which we intend to address in this paper: publishing mathematics using these representations is not attractive for the everyday mathematician. There is ongoing work to improve this state of facts, the latest the authors are aware of being presented in [15].

The mathematical documents that a user types into the computer are one main ingredient in building a mathematical knowledge base. In the ideal case, the human user does nothing else than typing his or her ideas and formulae into the computer, using a user-friendly environment that allows easy formula editing, like Maple or Mathematica. A program will take, then, this document and process it, extracting all the information of interest, organizing it, correlating it with (eventual) existing documents, making it available for the theorem provers, maybe even correcting eventual typos. As this is at the moment not yet possible, we may try to come as close as possible to such an mathematical authoring environment. For this, we have to ask the user to accept the current limitations of the existing computer programs and follow some well thought guidelines in writing the documents.

The main goal of the mathematical document editing environment we propose is to let the author concentrate on writing. We want to reduce the task of semantically annotating the document the user is working on to a minimum necessary. In order to fruitfully process the finished document we restrict the author to use a certain style for it. Most importantly, the user should:

A. separate text from mathematical formulae; and
B. group the formulae under certain headers (Definitions, Theorems, Propositions, etc.).

When a document respects the A. and B. requirements, a purpose specific document parser is able to

- identify the mathematical content from the rest of the document,
- correctly identify the mathematical knowledge types of the formulae, and
- store the identified knowledge in a form that is usable for other automated activities, e.g. proving.

We envision that advanced tools will take the output of such a dedicated document parser and extract more information from it, like singling out the defined concepts and their properties, generate new knowledge, etc.

3 Environment Description

We believe that there are certain actions that have to be performed from the moment a user decides to write a document with a mathematical content to the point where the document becomes a part of a mathematical knowledge base. We identified three such actions: a) writing the document following some guidelines; b) verifying (parsing) the document; and c) inserting the document into the knowledge base. Which guidelines we mean at point a) will become clear in Subsection 3.1. In the following, we discuss how each of these actions is performed in the environment proposed.

The implementation of our ideas is done in the frame of Mathematica and *Theorema*. The *Theorema* system is designed to assist a mathematician in all of the phases of his or her work (see [3, 5]). It is built on top of the computer algebra system Mathematica [27]. As a mathematical editing environment, Mathematica offers a very good front end support by giving the possibility of combining text, mathematical expressions, graphics, code in the same document, called “notebook”.

Theorema already provides constructors for writing, using, and composing formal, mathematical basic knowledge (**Definition**, **Proposition**, **Theory**, etc.). However, only few attempts wore done in building a base of formal mathematical knowledge in a systematic way, a knowledge base that can be browsed, extended and used for proving or teaching.

The environment we are about to describe is intended to improve this. With this purpose in mind, we have designed a special Mathematica stylesheet and implemented a set of functions for processing the notebooks that make use of it. We will refer to this environment as the "theory development environment".

To start working within the theory development environment the user has to open *Theorema*'s “Library Utilities” palette. This can be done, with the *Theorema* system loaded, by calling **OpenLibraryUtilities**[ ]. The functionality of the buttons on this palette will gradually be explained in the following subsections.

3.1 Writing the Document

To write a document that is to be included in a mathematical knowledge base, the author has to use a certain type of notebook. This will ease the annotation part of the work when typing the document into the computer. The document type we ask to be used employs the stylesheet facilities of Mathematica. A Mathematica stylesheet is a special kind of notebook, defining a set of styles that are allowed to be used in another notebook ([27] section 2.10). As mentioned before, we have defined a special stylesheet that allows annotating the document a user is working on, without his or her explicit awareness. The annotation is done while writing and is not semantic: it only marks cells and groups of cells in the notebook. This stylesheet will facilitate the parsing of the finished document.

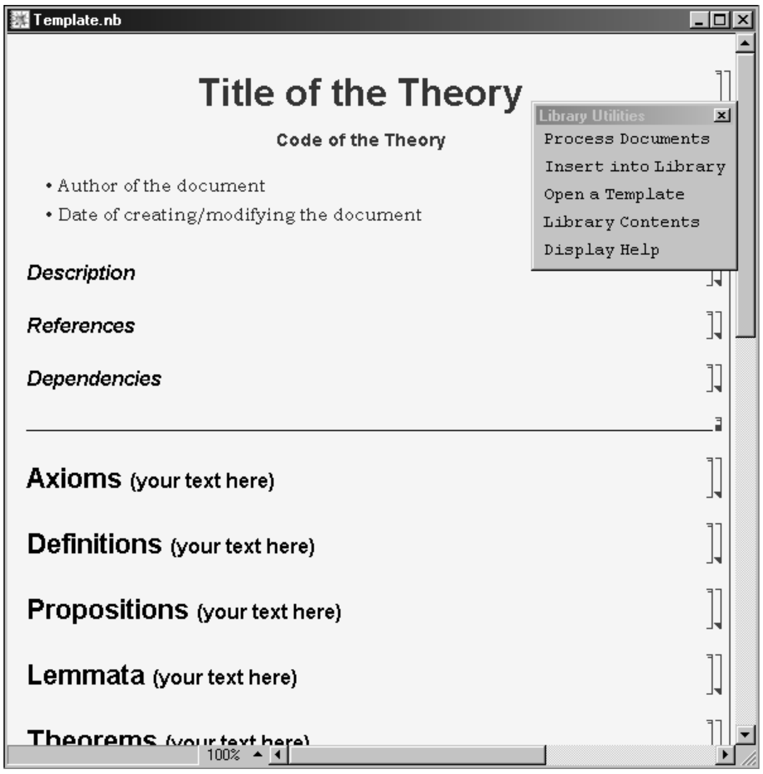
The simplest way to get a document with the specific style sheet is to use the 'Open a Template' button on the “Library Utilities” palette. What we obtain is a document like in Figure 1. (The figure also presents the “Library Utilities” palette).

The users that are acquainted with the Mathematica front end can also proceed differently, by opening a new notebook and choosing the ‘TheoremaTheory’ stylesheet for it. We will continue our description with the assumption that the user pressed the suggested button on the palette and has now opened a notebook like in Figure 1, which we will call ‘the theory notebook’ from now on.

The theory notebook is divided into two parts: header and content.

The header part of the theory notebook contains a title and a code, an author, a description, a reference and a dependencies section.

The code cell contains a short string of characters that is associated with the notebook and its content. The user is not compelled to type in a code, though he or she may prefer one that is a kind of compression of the document's title. The reason for



Title of the Theory

Code of the Theory

- Author of the document
- Date of creating/modifying the document

Description

References

Dependencies

Axioms (your text here)

Definitions (your text here)

Propositions (your text here)

Lemmata (your text here)

Theorems (your text here)

Figure 1: Theory notebook template.

this is revealed in the subsections below. When no code is given, one will be generated when the document is verified (subsection 3.2).

The author section is a text cell where the author of the document will put his or her name and the date the file was created.

The description section is reserved for, as its name says, describing in a few sentences what the content of the document is. The author can add here more information about the mathematical insights that a human reader may expect to get when reading the document.

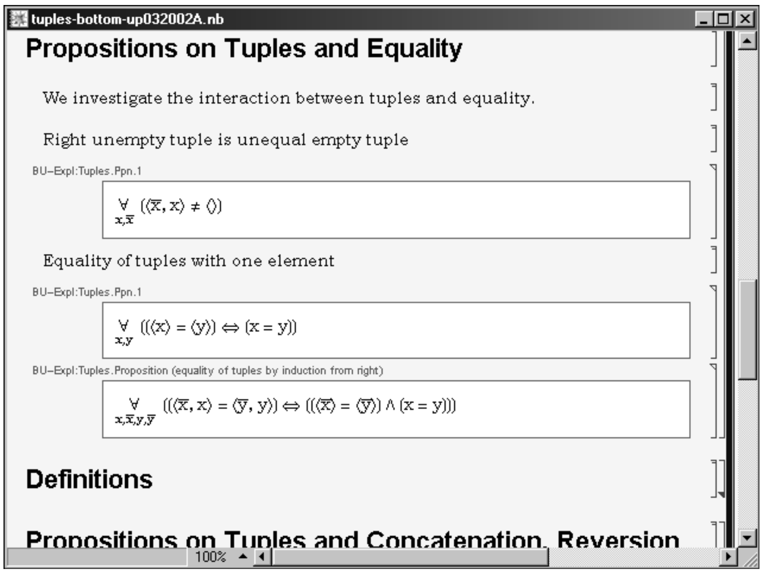
In the reference section the author can add pointers to books, web addresses, etc. from where the document content was gathered or where more information can be obtained. The author is free to add other information as well, leaving to his/her common sense that it is relevant for this section.

The dependencies section is giving the author the possibility to specify what other (existing) knowledge is needed in the current document. The author will have to specify, here, the codes of the used theories and the specification of the used parts, if this is the case. (For example, if the author wishes to use the axioms that define the real numbers, which are to be found in an existing document with the code **RealNos**, he or she has to write **Include**[**"RealNos.Axioms"**] in this section)

Only the document title is mandatory to be present in a theory notebook.

The content part of the document is where the actual formulae of the theory are to be typed in. The basic kinds of mathematical knowledge recognized are axioms, definitions, propositions, lemmata, theorems, corollaries, algorithms. The template document provides, for each of them, headings which, based on the style sheet definitions, will mark the formulae underneath them as axioms, definitions, etc. To make it easy to recognize the mathematical expressions we require that the formulae are typed in input cells. This does not put any burden on the authors, since it is the default cell type that will be considered as soon as one starts typing inside a Mathematica notebook.

For example, if a formula is considered to be a proposition it should be written under a heading with the style "Proposition". Though it contributes to clarity, it is not necessary that the word "proposition" appears in the text of the heading. The cell style of the heading has already the information that the formulae that will occur below this header will be propositions. The user can modify the header's text to better reflect the meaning of the formulae underneath it. The author is not restricted to only one section for a knowledge type. (see Figure 2)



tuples-bottom-up032002A.nb

Propositions on Tuples and Equality

We investigate the interaction between tuples and equality.

Right unempty tuple is unequal empty tuple

BU–Expl:Tuples.Ppn.1

$\underset{x,\overline{x}}{\forall} (\langle \overline{x}, x \rangle \neq \langle \rangle)$

Equality of tuples with one element

BU–Expl:Tuples.Ppn.1

$\underset{x,y}{\forall} ((\langle x \rangle = \langle y \rangle) \Leftrightarrow (x = y))$

BU–Expl:Tuples.Proposition (equality of tuples by induction from right)

$\underset{x,\overline{x},y,\overline{y}}{\forall} ((\langle \overline{x}, x \rangle = \langle \overline{y}, y \rangle) \Leftrightarrow ((\langle \overline{x} \rangle = \langle \overline{y} \rangle) \wedge (x = y)))$

Definitions

Propositions on Tuples and Concatenation, Reversion

Figure 2: Propositions in a theory notebook.

If the author wishes to attach labels to formulae this can be easily done by adding a tag to the cell where they occur. In the document, they will appear in a smaller font just above the formula cell. Tagging cells is a feature of the Mathematica front end (see [27] section 2.11.9). After verifying the document each formula will have a label attached. The labels are used to uniquely identify a formula in a library of theories.

As a final remark to this subsection we mention that the user can add anywhere in the document textual information that helps a human reader understand the presented knowledge.

3.2 Verifying the Document

Starting the verification process is done by pressing the 'Process Documents' button on the "Library Utilities" palette. The stylesheet used for writing the document helps identifying within it the information that is of interest for further processing.

The first step in verifying the document is to check wether the theory notebook has a title and a code. If the title is missing the verifying process stops with an error message. If the code is not present in the theory notebook the verifier will compute one by taking the first letters of the words appearing in the title, and will add it in the notebook. The verifier will check now the theory code against a list of existing theory codes that it has extracted from the knowledge base. If there is a name clash an error message is returned and the process stops. The author has to correct this problem.

Next, the document verifier will check that the theories and part of theories that are mentioned in the dependencies section are valid and do not lead to circles in the dependencies graph. If there is a loop detected the process stops with an error message and the user has to correct this matter.

Having passed these checks the verifiers will generate and attach labels to the formulae in the file. The generation takes into account the theory code, the knowledge type (axiom, definition, etc) and a numeric counter. This combination will uniquely identify the formula among all the formulae in the knowledge base. When a formula already has a user-given label, the verifier will not generate a label for it, but it will add the theory code in front of it. Figure 2 presents a part of a verified theory notebook.

In the end, the verifier will also add a content section in the header part of the document. This section is a compressed image of the content part of the document, having hyperlinks to formulae in it. This is meant to help a human reader to find a formula by just a click on its label.

3.3 Inserting the Verified Document into the Library

The verification process described above can be performed several times. When no errors occurred, the theory document can be inserted into the theory library. This is done by pressing the 'Insert into Library' button on the "Library Utilities" palette.

The procedure will use *Theorema*'s input parsing routines for the mathematical formulae that occur in the document. Each of the formulae will be read, parsed and the proper *Theorema* constructs will be created for it. This is the moment where the annotations made via the style sheet used for editing the document play an important role. A Mathematica package file, that contains the *Theorema* constructs, is created. Loading this package will make available to the *Theorema* system all the formulae that were introduced in the theory notebook. They can be used in the proving process.

At the same time, an entry about the theory notebook is made in a special theory index file. The theory index file keeps a record of each theory notebook that is part of the theory library. This includes information on where the file and its corresponding Mathematica package are stored.

The functionality of the 'Library Contents' button on the "Library Utilities" palette is the following: based on the entries stored in the theory index file, it will dynamically construct and present the user a notebook with a list of theories already existing in the knowledge library. The list has hyperlinks to the notebooks where the theories are introduced.

4 Concluding Remarks and Future Work

We have presented an environment for editing documents, verifying and including them into a mathematical knowledge library. This environment allows the users to concentrate on writing, requiring only that they use a certain style sheet for their documents. A document that uses this style sheet can be automatically processed in order to extract its mathematical content and store it in a format that can be used for browsing, proving, etc. For example, we could apply the tools described in [22] for obtaining derived knowledge.

The theory library that is built using the described environment comprises both the documents written by the authors and the processed files obtained out of them. The reason for this is that a human reader will want to read and inspect the former, while an automated theorem prover will use the latter.

There are features that are missing in our environment and are subject to future work. Among them we mention the plan to improve the routine that extracts the mathematical content from the theory notebook and inserts it into the theory library. For example, automatically identifying the defined symbols in the document should be possible, the user should be allowed to hierarchically organize the formulae in the theory notebook. Also, we did not yet thoroughly consider how searching for notions and concepts can be done best in such a theory library. Another issue is how to manage modifications that the user might perform to the documents that are already included in the theory library.

References

[1] S. Allen, M. Bickford, R. Constable, R. Eaton, C. Kreitz, L. Lorigo. *FDL: A Prototype Formal Digital Library.* Cornell University, 2002. (http://www.nuprl.org/FDLproject/02cucs-fdl.html)

[2] A. Asperti, B. Buchberger, J.H. Davenport, James Harold (Eds.) Proceedings of the Second International Conference, MKM 2003 Bertinoro, Italy, February 16-18, 2003 Series: Lecture Notes in Computer Science, Vol. 2594, 2003, X, 225 p. Also available online. Softcover ISBN: 3-540-00568-4

[3] B. Buchberger. *Mathematical Knowledge Management Using Theorema.* In [11].

[4] B. Buchberger, G. Gonnet, M. Hazewinkel. *Annals of Mathematics and Artificial Intelligence*, Volume 38, Volume 38, Number 1-3, May 2003. Kluwer Academic Publishers, ISSN 1012-2443.

[5] B. Buchberger. *Theorema: A short introduction.* The Mathematica Journal, 8(2):247–252, 2001.

[6] B. Buchberger. *Algorithm Invention and Verification by Lazy Thinking.* In: D. Petcu, V. Negru, D. Zaharie, T. Jebelean (eds), Proceedings of SYNASC 2003 (Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, October 14, 2003), Mirton Publishing, ISBN 9736611043, pp. 226.

[7] B. Buchberger. *Groebner Rings in THEOREMA: A Case Study in Functors and Categories*, SFB (Special Research Area) "Scientific Computing" Technical Report Nr. 2003 - 46, Johannes Kepler University, Linz, Austria, 2003.

[8] B. Buchberger. *Towards the Automated Synthesis of a Grbner Bases Algorithm.* RACSAM (Review of the Royal Spanish Academy of Science), to appear, 10 pages.

[9] B. Buchberger. *Computer-Supported Mathematical Theory Exploration: Schemes, Failing Proof Analysis, and Metaprogramming.* Submitted, also available as Technical Report from RISC, Johannes Kepler University, Austria.

[10] B. Buchberger, A. Craciun. *Algorithm Synthesis by Lazy Thinking: Examples and Implementation in Theorema.* In: Fairouz Kamareddine (ed.), Proc. of the Mathematical Knowledge Management Workshop, Edinburgh, Nov. 25, 2003, Electronic Notes on Theoretical Computer Science, volume dedicated to the MKM 03 Symposium, Elsevier, ISBN 044451290X, to appear.

[11] B. Buchberger, O. Caprotti. Editors of the Proceedings of the First International Workshop on Mathematical Knowledge Management: MKM'2001 RISC, A-4232 Schloss Hagenberg, September 24-26, 2001. ISBN 3-902276-01-0.

[12] B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger. *The Theorema Project: A Progress Report.* In: Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6–7, 2000, St. Andrews, Scotland, M. Kerber and M. Kohlhase eds.), A.K. Peters, Natick, Massachusetts, pp. 98–113. ISBN 1–56881–145–4.

[13] O. Caprotti, D. Carlisle. *OpenMath and MathML: Semantic Mark Up for Mathematics.* In ACM Crossroads, ACM Press, 1999.

[14] *The Coq proof assistant.* (http://coq.inria.fr/coq-eng.html)

[15] G. Goguadze, A. González Palomo. *Adapting Mainstream Editors for Semantic Authoring of Mathematics.* Presented at the Mathematical Knowledge Management Symposium, November 2003, Heriot-Watt University, Edinbourgh, Scotland.

[16] *IMPS: An Interactive Mathematical Proof System.* Developed at The MITRE Corporation by W. M. Farmer, J. D. Guttman, F. J. Thayer. (http://imps.mcmaster.ca/)

[17] *Isabelle.* Developed at Cambridge University (Larry Paulson) and TU Munich (Tobias Nipkow). (http://www.cl.cam.ac.uk/Research/HVG/Isabelle/index.html)

[18] M. Kohlhase. *OMDoc: An Infrastructure for OpenMath Content Dictionary Information.* In ACM SIGSAM Bulletin, volume 34, number 2, pages 43-48, 2000.

[19] M. Kohlhase, A. Franke. *MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems.* Journal of Symbolic Computation 23:4 (2001), pp. 365 – 402.

[20] D.W. Lozier, *NIST Digital Library of Mathematical Functions.* In Annals of Mathematics and Artificial Intelligence, vol. 38, No. 1–3, May 2003. Eds. B. Buchberger, G. Gonnet, M. Hazewinkel. Kluwer Adacemic Publishers, ISSN 1012-2443.

[21] *The Mizar System.* Developed at the University of Warsaw, directed by A. Trybulec. (http://mizar.uwb.edu.pl/system/)

[22] K. Nakagawa, B. Buchberger. *Two Tools for Mathematical Knowledge Management in Theorema.* In [11].

[23] S. Owre, J. Rushby, N. Shankar, D. Stringer-Calvert, *PVS: An Experience Report*, In: Applied Formal Methods—FM-Trends 98. Eds. D. Hutter, W. Stephan, P. Traverso, M. Ullman. LNCS vol. 1641, pp. 338–345. Springer-Verlag, Germany.

[24] F. Piroi. *Tools for Using Automated Provers in Mathematical Theory Exploration.* Ongoing PhD thesis, to be finished in autumn 2004.

[25] S. Rockey. Mathematics Digitization, at Cornell University, Mathematics Library. http://www.library.cornell.edu/math/digitalization.php

[26] W3C Math Home: What is MathML? (http://www.w3.org/Math/)

[27] S. Wolfram. *The Mathematica Book.* Wolfram Media Inc. Champaign, Illinois, USA and Cambridge University Press, 1999.