An Environment for Building Mathematical Knowledge Libraries

Florina Piroi and Bruno Buchberger

Research Institute For Symbolic Computation,
4232 Hagenberg, Austria
{Florina.Piroi, Bruno.Buchberger}@risc.uni-linz.ac.at

Abstract. Proving is an activity that makes use of mathematical knowledge. For a human, this knowledge - together with the know-how about proving techniques - is accumulated over years of studying mathematics. For a theorem proving assistant, the mathematical knowledge needed in the proving process has to be provided beforehand. In this paper we describe a user-friendly environment for building up a mathematical knowledge base that can be accessed by an automated proving assistant.

1 Introduction

Whether it is done by a human or by an automated system, proving is a high-level activity that heavily uses already defined concepts and proven facts. In the case of a human prover, years spent on studying different fields of mathematics decisively contribute in building up a kind of a mathematical knowledge base. This contains axioms, theorems, proofs, etc. as well as proving techniques, that the mathematician is using while reasoning about something, at times even unaware of it.

In the field of Automated Theorem Proving a lot of research is done on "teaching" the machine how to prove, by translating knowledge that is implicit for a human mathematician (namely proving techniques and methods) to an explicit form, that a machine can understand and use. Though this direction of research is by no means exhausted, there is now an increased need of having computer accessible mathematical knowledge bases.

In the last years significant steps were done in this direction and the importance of mathematical knowledge management was first underlined in 2001, at the First International Workshop on Mathematical Knowledge Management. In [2] are identified three main problems of the mathematical knowledge management area, namely:

- retrieving mathematical knowledge;
- building up mathematical knowledge bases; and
- educating mathematicians to work efficiently with and improve the existing knowledge bases.

Before we can speak about managing a mathematical knowledge base we first have to have a mathematical knowledge base. At the moment, the largest such base is Mizar [13]. Among other existing ones we mention here MBase [12], the Formal Digital Library project [1], the NIST Digital Library of Mathematical Functions [15], the libraries of the theorem provers Isabelle [10], PVS [18], IMPS [9], Coq [7].

This paper presents an environment for editing and processing mathematical documents, making them available for including them in a mathematical knowledge base. Ultimately, we want to use the knowledge stored in such a knowledge base in automated proving. The ideas that originated this work are due to the second author, elaborating them and the implementation is done by the first author. This work will also appear as part of [17].

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 idea of the design of our mathematical document editing environment. In Section 3 we will describe how this is done in the frame of *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.

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 [19], it is now possible to display and communicate formulae. Being an application of XML, MathML benefits of 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 [6] and OMDoc [11] 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. At a first glance, 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 [8].

The main ingredient in building a mathematical knowledge base are the documents with mathematical content, that a user types into the computer. This is, usually, a time consuming activity that is not always eased by the document editing environment. In the ideal case, the human user does nothing else than typing her ideas and formulae in 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, translating it into a form that is understood by 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 a 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-out guidelines for 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

The main phases that a document goes through, from the moment a user decides to write it (i.e. type it into the computer), to the point where the document

becomes part of a knowledge base, are: a) writing the document following some guidelines; b) verifying (parsing) the document; and c) inserting the document into the knowledge base. 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 her work (see [2, 3]). It is built on top of the computer algebra system Mathematica [20]. 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. Another feature of Mathematica that is used by the *Theorema* system and is relevant for our environment is that Mathematica permits programming syntax for new symbols.

We note here, that all formal knowledge in the documents written using this environment are predicate logic formulae so that their semantics can be fully explained within the document. For this reason, translating the output of the document parser to a format like OMDoc or MathML can be done relatively easy ([17]). The reverse translation is not always possible because the content of MathML, OMDoc documents are not fully in predicate logic.

Theorema already provides constructors for writing, using, and composing mathematical basic knowledge **Definition**, **Proposition**, **Theory**, etc.). However, there were few attempts for building up a mathematical knowledge library in a systematic way, a library 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 are making use of this stylesheet. We will refer to this environment as the "theory development environment".

Belonging to the theory development environment is a *Theorema* palette (similar to a toolbar), named "Library Utilities", that can be opened by calling **OpenLibraryUtilities**[ ], after the *Theorema* system is loaded. 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 stylesheet for it. This will ease the annotation part of the work put in writing. The document type we ask to be used employs the stylesheet facilities of Mathematica. A Mathematica stylesheet is a special kind of notebook that defines a set of styles which are allowed to be used in another notebook ([20] Section 2.10). As mentioned before, we have defined a special stylesheet that allows annotating the document a user is working on, without 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 style sheet will facilitate the parsing of the finished document.

The simplest way to open a new document with the specific stylesheet 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

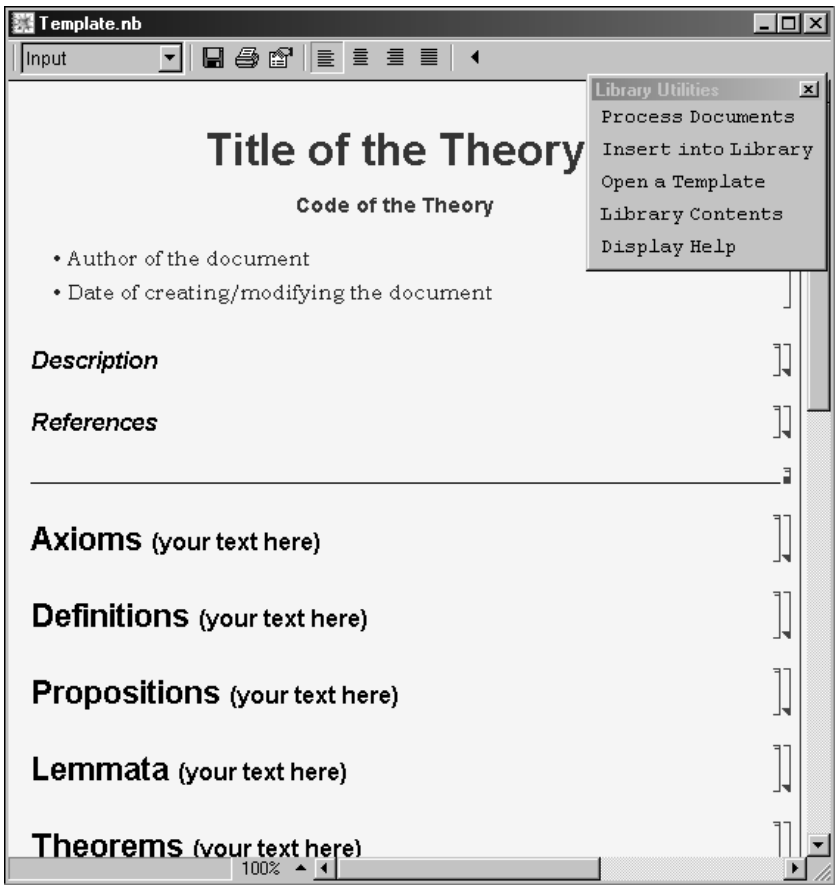

Fig. 1. Theory notebook template.

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 and a reference 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 she may prefer one that is a kind of compression of the document's title. When no code is given, one will be generated when the document is verified (Subsection 3.2). The code of the document will, later, uniquely identify it (and its content) among other documents in the theory library.

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

The description section is reserved for, as its name says, describing in a few sentences, informally, 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 her common sense that it is relevant for 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 stylesheet's definitions, will mark the formulae underneath them as axioms, definitions, etc. For making 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. As an example, if a formula is considered to be a proposition it should be appear 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).

If the author wishes to attach labels to formulae this can be easily done by adding a tag to the cell where they occur. Labels are Mathematica strings and, in the document, will appear in a smaller font just above the formula cell. Tagging cells is a feature of the Mathematica front end (see [20] section 2.11.9). It is also possible to give labels to groups of formulae by tagging the heading under which they are written. Labels can be used to indicate that the theory being written uses (formulae and/or parts of) other theories. For example, if the author wishes to indicate that the definitions of tuples found in the theory document with the code "BU–Expl:Tuples", are assumed in the document being written, he or she has to use **Include**["BU–Expl:Tuples.3"], where "BU–Expl:Tuples.3" is the label attached to the set of definition formulae. The **Include**[**...**] directives can be placed anywhere in the content part of the mathematical document.

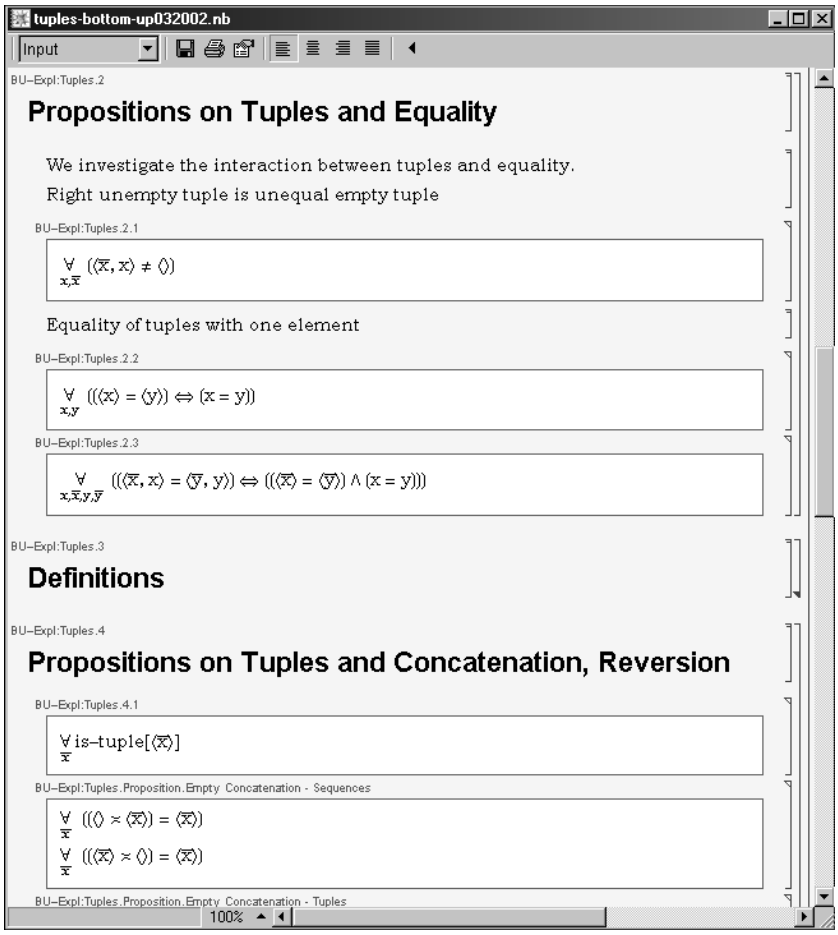

Fig. 2. Propositions in a theory notebook.

Labels also give the author the possibility to group formulae into smaller theories, composing them hierarchically if wished. An example of how this can be done is shown in Figure 3. The cells that contain the theory definitions must be under a header with the style "Theory" (provided by the used stylesheet) so that the document verifier can treat them correspondingly. 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 Consistent Parsing of Documents

Starting the parsing process is done by pressing the 'Process Documents' button on the "Library Utilities" palette. The stylesheet used for writing the document

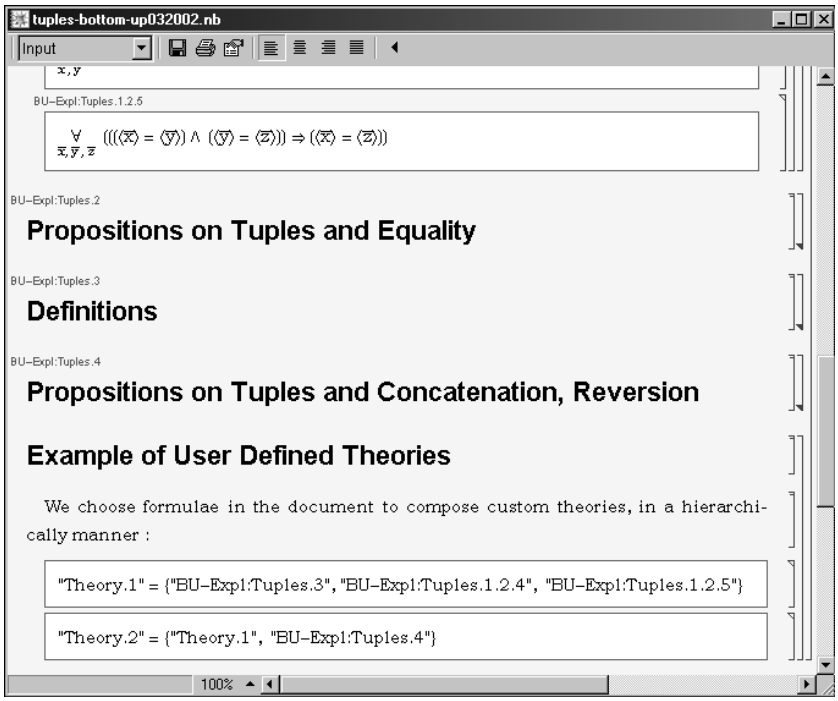

Fig. 3. User defined theories.

helps identifying within it the information that is of interest for further processing.

The first step in parsing the document is to check wether the theory notebook has a title and a code. If the title is missing the parsing process stops with an error message. If the code is not present in the theory notebook, the parser will compute one by taking the first letters of the words appearing in the title, and will add it in the notebook. The parser will check then the theory code against a list of existing theory codes that it has extracted from the knowledge library. 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 parser will generate labels for each of the formulae cells and for their headings. The generation takes into account the theory code and numeric counters. These labels will uniquely identify the formulae 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. Figures 2 and 3 present parts of a verified theory notebook.

Now, the document parser will check whether the theories and/or part of theories that are referred to via **Include**[**...**] commands exist in the mathematical theory library. It will also check whether these theories, at their turn,

include (parts of) other theories and that this chain of inclusions does not lead to cycles. If there is a loop detected the process stops with an error message and the user has to correct this matter.

The document parser 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, with hyperlinks to formulae in it. Additionally, hyperlinks to the included theories are created.

Lastly, the parsing procedure will apply *Theorema*'s input parsing routines on the mathematical formulae that occur in the document. Each of the formulae will be read, parsed and, based on the annotations carried out by the stylesheet used, the proper *Theorema* constructs will be created for it. *Theorema* theory constructs are created for the user defined theories (if any). Furthermore at this stage, *Theorema* theory constructs collecting the formulae underneath the main headers of the theory notebook are created for each of the headers.

A Mathematica package file, that contains the *Theorema* constructs for the document being parsed, is created. Loading this package will make all the formulae that were introduced in the theory notebook available in the *Theorema* system. They can be used in the proving process.

In later developments of the environment, the annotations made via the stylesheet used for editing the document may play an important role in attaching a semantical meaning to the formulae in the document by triggering further, knowledge-type specific analysis of the logical formulae.

3.3 Inserting the Verified Document into the Library

The document parsing 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 verified documents will be copied into the theory library location, together with the created Mathematica package. At the same time, an entry about the theory notebook is made in a special theory index file. The theory index file is keeping record of each theory notebook that is part of the theory library. This includes the theory code, information on where the file and its corresponding Mathematica package are stored, and theories on which the inserted one depends.

From the moment the document is inserted into the theory library its content is available for inclusion, via **Include**[**...**] commands, in other documents that the author intends to write.

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. The user is also provided with commands that help her inquire the content of a theory. She can ask, for example,

which are the axioms of a theory with a specific code. The system will return a list of labels of the axioms that are present in the respective theory.

4 Concluding Remarks and Future Work

We have presented an environment for editing documents with mathematical content, parsing and including them into a mathematical knowledge library. This environment is designed such that the users can concentrate on writing their ideas, requiring only that they use a certain stylesheet for their documents. A document that uses this stylesheet 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 [16] for obtaining derived knowledge.

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. 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.

The theory library that is built using the described environment comprises both the documents written by the authors and the processed files obtained from 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. The format of the processed documents is understood by the *Theorema* system, in the sense that we can apply the various reasoners of *Theorema* to this knowledge (for proving, simplifying, etc).

Applying stylesheets to documents with mathematical contents has already been used before, though with a different scope than ours. For example, in [14] automatically generated stylesheets are used for displaying the mathematical knowledge in different forms.

In our view, logical manipulation of mathematical knowledge (i.e. mathematical theory exploration) does not yet get sufficient attention in the area of mathematical knowledge management. In *Theorema*, however, this is the main focus. By logical manipulation of mathematical knowledge we mean:

- invention of concepts;
- invention and verification of propositions;
- invention of problems;
- invention and verification of algorithms;
- storage of mathematical knowledge (concepts, propositions, problems, algorithms) for later easy retrieval by logical analysis.

In [4] it is described how all this can be organized in exploration cycles according to a "four thread model".

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. B. Buchberger. *Mathematical Knowledge Management Using Theorema.* In Proceedings of the First International Workshop on Mathematical Knowledge Management: MKM'2001 RISC, A-4232 Schloss Hagenberg, September 24-26, 2001. Eds. Bruno Buchberger and Olga Caprotti. ISBN 3-902276-01-0
3. B. Buchberger. *Theorema: A short introduction.* The Mathematica Journal, 8(2):247–252, 2001.
4. B. Buchberger. *Computer–Supported Mathematical Theory Exploration: Schemes, Failing Proof Analysis, and Metaprogramming.* Submitted to AISC 2004.
5. 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.
6. O. Caprotti, D. Carlisle. *OpenMath and MathML: Semantic Mark Up for Mathematics.* In ACM Crossroads, ACM Press, 1999.
7. *The Coq proof assistant.* (http://coq.inria.fr/coq-eng.html)
8. 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.
9. *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/)
10. *Isabelle.* Developed at Cambridge University (Larry Paulson) and TU Munich (Tobias Nipkow). (http://www.cl.cam.ac.uk/Research/HVG/Isabelle/index.html)
11. M. Kohlhase. *OMDoc: An Infrastructure for OpenMath Content Dictionary Information.* In ACM SIGSAM Bulletin, volume 34, number 2, pages 43-48, 2000.
12. 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.
13. *The Mizar System.* Developed at the University of Warsaw, directed by A. Trybulec. (http://mizar.uwb.edu.pl/system/)
14. B. Naylor, S. Watt. *Meta-stylesheets for the conversion of mathematical documents into multiple forms.* 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.
15. 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.
16. K. Nakagawa, B. Buchberger. *Two Tools for Mathematical Knowledge Management in Theorema.* In Proceedings of the First International Workshop on Mathematical Knowledge Management: MKM'2001 RISC, A-4232 Schloss Hagenberg, September 24-26, 2001. Eds. Bruno Buchberger and Olga Caprotti. ISBN 3-902276-01-0
17. F. Piroi. *Tools for Using Automated Provers in Mathematical Theory Exploration.* Ongoin PhD thesis, to be finished in autumn 2004.

18. 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.
19. W3C Math Home: What is MathML? (http://www.w3.org/Math/)
20. S. Wolfram. *The Mathematica Book*. Wolfram Media Inc. Champaign, Illinois, USA and Cambridge University Press, 1999.