Johann Radon Institute for
Computational and Applied Mathematics
Austrian Academy of Sciences (ÖAW)

Label Management in Mathematical Theories

B. Buchberger, F. Piroi

RICAM-Report 2004-16

Label Management in Mathematical Libraries

Florina Piroi* and Bruno Buchberger**

Florina.Piroi@oeaw.ac.at
Bruno.Buchberger@risc.uni-linz.ac.at

*Johann Radon Institute
for Computational and Applied Mathematics (RICAM),
Austrian Academy of Sciences,
Altenbergerstrasse 69, 4040 Linz, Austria

and

**Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz,
Schloss Hagenberg, 4232 Hagenberg i.M., Austria

Abstract. In this paper we identify the organizational problems of Mathematical Knowledge Management and describe tools that address one of these problems, namely, the management of composite, hierarchical labels for formalized knowledge. We describe how the tools are realized in the frame of the *Theorema* system.

Acknowledgments This work was partially supported by the RISC PhD scholarship program of the government of Upper Austria, by the FWF (Austrian Science Foundation) Spezialforschungsbereich F013, Numerical and Symbolic Scientific Computing, project P1302, and by the RICAM Institute of the Academy of Sciences, Linz. We also thank the anonymous referees who, with their comments, motivated us to clarify the position and value of the content of this paper in the frame of the overall Mathematical Knowledge Management (MKM) research area by explicitly describing our view on MKM and its subareas.

1 Introduction

1.1 Label Management as Part of Mathematical Knowledge Management

In order to specify the exact goal of the tool described in this paper we first describe our view of the new research area Mathematical Knowledge Management (MKM) and its subareas. Roughly, this view was expressed by the second author in the preface of the first conference on MKM and the subsequent special issue of the journal AMAI, see [10] and [13]. Within this view of MKM, the current paper only deals with subarea c2., see below. For other, alternative, views of MKM see the introductions of recent papers on MKM, in particular the ones in [10, 3] and [13].

In our view, the aim of MKM is the computer–support (partially or fully automated) of all phases of the mathematical theories exploration:

- 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 they can be easily accessed and applied at a later time.

MKM in this broad sense is, essentially, a logical activity: All formulae (axioms, definitions of 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), i.e. reasoning guided by explicit algorithmic rules.

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 the logical aspects of MKM within *Theorema* are [12, 6]. 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 [7, 9] and [11]. The formal (computer–supported) reasoning aspect of MKM is not a goal of this paper.

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 computers: This is the problem of "digitizing" mathematical knowledge, see e.g. [35] for a survey on the existing projects in this area or [15]. The *Theorema* project is not engaged in this area of MKM.
b. The translation of digitized mathematical knowledge, for example in the form of LaTeXfiles, 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 theorem proving assistants): Many current projects are addressing this question, see e.g. MathML [36], OpenMath [14]. The *Theorema* project is not engaged in this area of MKM either. In fact, we think that most of the mathematical papers, even if their formulae are typed in LaTeX, are logically not sufficiently consistent and explicit for automated extraction of their logical content. Therefore, in our own experiments on formalization of mathematical theories, we prefer to build–up mathematical theories by radical re–formalization from scratch. Such re–formalizations may well follow the general flow of presentation in an existing paper or textbook but the actual formulation of the formulae has to be done "by hand" or by formal reasoning tools.

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 [31]. Among other existing ones we mention MBase [27], the Formal Digital Library project [1], the NIST Digital Library of Mathematical Functions [30], Hypertextual Electronic Library of Mathematics [20], the libraries of the theorem provers Isabelle [22], PVS [32], IMPS [21], Coq [17].
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. *Theorema* puts a particular emphasis on this aspect, see for example, [8]. However, this aspect is not treated in this paper either.
   c2. The additional assignment of various kinds of labels to formulae and collections of formulae so that blocks of mathematical knowledge can be identified and combined in various ways without actually going into the "semantics" of the formulae.

This paper exclusively deals with subproblem c2. In traditional mathematical texts, of course, labels are used but a systematic management of labels is, normally, not considered to be important nor is it feasible. In contrast, in the build–up of completely formalized mathematical knowledge bases, the systematic design and processing of structured labels (i.e. individual labels like "(1)", "(2)" or "(associativity)" etc., hierarchical section headings, key words like "definition" and "theorem", names of files etc.) becomes vital for the automated structuring and re–structuring of collections of formulae as input to formal reasoning tools like provers, simplifiers, algorithm verifiers, model checkers etc.. Consequently, we need algorithmic tools that handle all types of labels and allow us to partition and combine, structure and re–structure mathematical knowledge bases according to the structural information provided by the hierarchical labels.

In order to avoid misunderstandings, let us emphasize that, in our view, labels do not intend to have any logical meaning or functionality. This is in contrast to the goal of "annotations", etc. as, for example, in [36], [14], and [26], which convey at least part of the semantics. In our view, the semantics of formulae (in particular predicate logic formulae) is exclusively defined by their inclusion into the context of collection of other formulae (mathematical knowledge bases). In other words, formulae obtain their meaning relative to each other in the context of the knowledge base in which they occur and in the context of the logic used for reasoning about the formulae, whereas labels only help in addressing, referencing, selecting individual formulae in knowledge bases and in partitioning and re–combining (small and big) collections of formulae. Summarizing, in the view of this paper, the functionality of labels is purely organizational and not logical.

Also, the concept of labels has to be distinguished from the concept of "comments". Comments have neither a logical meaning nor do they contribute to the organization of mathematical knowledge bases. Rather, comments are only meant as meta–level guides for human readers of mathematical knowledge bases.

Comments are actually skipped in the algorithmic (logical and organizational) processing of knowledge bases. Hence, from the point of view of MKM, comments are trivial and we do not say anything about them in this paper. In fact, within the *Theorema* system, there is ample possibility for comments: Since *Theorema* uses the front–end of Mathematica, in *Theorema* files comments can be put everywhere into Mathematica "text cells" and are just overread in any processing of the files.

Before we go into technicalities let us add a remark on the notion of "working mathematicians", who are the intended users of *Theorema* and similar systems. Often, people criticize that "working mathematicians" do not think and work in predicate logic and that, therefore, systems like *Theorema* will not have any relevance for the working mathematicians. Rather, these people request that future systems for the "working mathematicians" should be made "more intelligent" in order to find out the "natural intended meaning" of the "usual informal texts" working mathematicians use to read and produce. We strongly oppose this view. In fact, the second author considers himself a working mathematician and, as a result of forty years of "working in mathematics", he felt and is feeling the strong need to develop systems for formal mathematics based on a well–defined logic (syntax and reasoning mechanisms) in order to make significant progress in the content, reliability, accessibility, re–usability, algorithmization, didactics, etc. of mathematics and also in the speed, efficiency and reliability of the process of "producing" (inventing and verifying) mathematical results. Also, as a matter of experience, it must be noticed that the formal quality of many current mathematical texts is quite low. Therefore, in our view, it is not worthwhile to develop "intelligent" tools for finding out the mathematical ingenuity that may be contained in mathematical papers that are unclear, partly inconsistent, badly designed, operating in unclear context, proposing unclear solutions for unclear problems, etc. Instead, we believe that time is better spent for educating the next generation of mathematicians in such a way that they have the formal ability to express their findings unambiguously, clearly and well structured in the frame of a formal language like predicate logic (in some attractive and easily manageable syntax). Of course, mathematicians should retain (and improve) their ability to express ingenious views in informal comments, which will always be an important part of "working in mathematics". With improved formal ability, "working mathematicians" will be able to enjoy the increasing support they can get from current and future MKM systems without giving up the artistic way of thinking by pictures, gestures, sketches of formulae, condensed symbols and "hand waving". In fact, since many years, the second author is teaching a special course devoted to the art of formal mathematics for students with background in mathematics or computer science. One of the main insights from this teaching is that increased formal training does not prevent people from being creative but, just to the contrary, frees the mind for the creative act of inventing and verifying mathematical insights in the same way as practising a musical instrument is the technical prerequisite for expressing creative musical ideas.

1.2 The Purpose of Label Management

Now let us describe the scenario that specifies the purpose and the functionality of the tools we designed and implemented, and which we present in this paper, for handling hierarchical labels in the *Theorema* system. This scenario will also make it clear how our tools can be used for any other MKM system that relies on predicate logic.

We start from the assumption that we treat collections of formulae in pure (higher order or first order) predicate logic in the internal form of nested expressions in prefix notation. For example,

$$\begin{array}{l}\texttt{TM}\texttt{ForAll}\,[\texttt{range}\,[\texttt{var}[f],\texttt{var}[B]],\texttt{True},\\ \qquad\texttt{TM}\texttt{Iff}\;\;[\texttt{is-bounded}[\texttt{var}[f],\texttt{var}[B]],\\ \qquad\qquad\texttt{TM}\texttt{ForAll}[\texttt{range}[\texttt{var}[x]],\texttt{True},\\ \qquad\qquad\quad\texttt{TM}\texttt{LessEqual}[\texttt{TM}\texttt{BracketingBar}[\texttt{var}[f][\texttt{var}[x]]],\texttt{var}[B]]]]]\end{array}$$

is such a formula. Collections of such formulae can either be input by a user "by hand" in the external syntax (see below), or they can be the result of some of the *Theorema* reasoning tools like provers, simplifiers, algorithm synthesizers, etc., or they can be the result of translating knowledge bases from any other MKM system (as long as these systems work in the frame of predicate logic).

Since the *Theorema* system is mainly meant as a practical tool for helping the working mathematician with exploring mathematical theories and presenting the trace and the result of theory explorations in an easy–to–read and easy–to–write style, we also provide an external form of predicate logic formulae. For example, the above formula, in the current standard external syntax of *Theorema* is as follows:

$$\forall_{f,B}\ (\texttt{is-bounded}[f,\ B] \Longleftrightarrow \forall_x\ |f[x]| \leq\ B).$$

This external syntax was carefully designed in order to come as close as possible to the "usual" syntax of mathematical formulae (including algorithms) in textbooks and articles. However, since "usual" syntax is a matter of endless dispute and heavily influenced by individual taste and practice, *Theorema* offers an extra tool which allows to program, within certain limitations, one's own external, two–dimensional syntax. Thus, if the user of *Theorema* does not like the external syntax provided as a default, she is welcome to design and implement a different one using the syntax programming tools of *Theorema*, which are actually provided by the underlying Mathematica system and by which formulae in the external syntax can be turned into the above internal standard syntax. (Flexible syntax programming was, in fact, one of the reasons why we decided to implement *Theorema* within Mathematica.)

Of course, it is also possible to type the variables appearing in the above example:

$$\forall_{f:\mathbb{R}\longrightarrow\mathbb{R},B\in\mathbb{R}}\ (\texttt{is-bounded}[f,\ B] \Longleftrightarrow \forall_{x\in\mathbb{R}}\ |f[x]| \leq\ B).$$

However, all this does not extend the class of predicate logic formulae and all these details of the logic language are not relevant for this paper.

We start now from the standard situation, in which we have a (small or big) collection of predicate logic formulae in the above *Theorema* syntax, contained in the input cells of a couple of *Theorema* files, which in fact are just ordinary Mathematica notebooks files. Such files have various kinds of "cells" (see [37], Section 1.3.5): Input cells, that contain formulae (in our case predicate logic formulae in *Theorema* syntax), text cells for comments (which have no relevance for the purposes of this paper), and a whole hierarchy of cells for "section headings" which, in this paper, we will use heavily for structuring collections of formulae: We consider headings as a kind of labels for whole blocks of formulae. For the purposes of *Theorema*, we added the possibility that formulae in input cells can have additional individual labels and, also, that formulae in input cells can be "wrapped" by additional key words like "definition", "theorem", "axiom", "algorithm", "lemma", "fact", etc. Note that these key words, again, are nothing else than a kind of labels: They do not at all add any logical meaning to the formulae they wrap and, actually, there is no way to decide whether a formula is a "definition" or a "theorem", etc., except by analyzing its role in the context of an entire theory exploration activity. In fact, a given formula may be a definition in one exploration situation and a theorem or an algorithm in some other exploration. The assignment of keywords like "definition" etc. is, hence, not something which is inherent in the formulae but is, rather, something that may change according to the view of the user who organizes collections of formulae and, therefore, is part of our labelling system and the tools we provide for managing labels.

The description of the tools we designed and implemented for managing hierarchic labels and the corresponding management of hierarchic collections of predicate formulae (in *Theorema* syntax) is the content of this paper. Typical users of these tools are "working mathematicians" who want to build up and explore mathematical theories within the *Theorema* system. However, by translators to and from other systems that process collections of predicate logic formulae, the tools we describe can also be used from within other systems. Currently, translators to and from *Theorema* collections of formulae are implemented for several deduction systems ([28]). More such translators are under way and will be added to the system depending on the available man power and user request. In particular, we are also working on translators between *Theorema* predicate logic formulae and OmDoc formulae. Alternatively, one could design and implement similar labeling management tools directly in other systems.

1.3 A Short Review of the Relevant Literature

It seems that label management in the sense specified in this paper is not an explicit goal in the current MKM systems. However, work on annotation of (collections of) formulae and formula editing has an overlap with and relevance for the work presented in this paper. Therefore, in this subsection, we give a short review on the pertinent papers.

Starting from the display of formulae as graphics in internet document, working W3C issued an MathML recommendation, [36], for displaying and communi-

cating formulae by means different from images. 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 [14] and OMDoc [26] emerged. The OpenMath standard concentrates on representing mathematical expressions together with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the world wide web. Essentially, 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 OpenMath objects used.

A 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, a task that can be 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, like the work described in [18] and the one in [25].

Furthermore, to our knowledge at the time of writing this paper, systems that use and/or manage big collection of documents with mathematical content do not make use of a document editing environment like the one we are about to describe. Within most of the proving systems, the users are typing their documents in an Emacs–based editor or something similar, see for example, Mizar [31], HOL [19], CoQ [17], PVS [32]. Where translators are provided, the files can be stored, later, in LaTeX, MathML or OpenMath formats. In this form, documents produced by proving assistants can be included in libraries of digital mathematics like HELM [20] and MBase [27]. (Helm includes part of the libraries of the proving assistants NuPRL [16] and Coq [17], MBase includes libraries from Omega [4] and TPS [2]). Systems that concentrate on representing and publishing mathematics (on the web) make use of document translators and formulae editing tools that translate formulae and documents to different formats. For example, the JOME OpenMath Editor [24] creates and manipulates OpenMath objects and within ActiveMath [29], jEditOQMath is a package of tools for editing and managing documents in OMDoc format [23].

1.4 Organization of the Paper

In the next section we will describe the problem of label management by an example. In Section 3 we present how we have realized the main routines of our label management tools, namely automatic generation of labels, use of formal knowledge to build new knowledge and use of knowledge as input to automated reasoners. In Section 4 we give some final comments on our work.

The design of the tools described in this paper 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 [33].

2 Problem Description by an Example

Let us take a look at the screen shots in Figures 1 and 2 of the appendix. They present part of the contents of two Mathematica notebooks storing text and formulae. The formulae are in predicate logic, in the *Theorema* external syntax. The formulae and the text in these files are grouped under certain headings in sections, subsections, etc. Such files can either be the result of an automated process, or can be created by a human user, via Mathematica's and *Theorema*'s front end environments. Typing the documents is done in a WYSIWYG style: Both Mathematica and *Theorema* provide several tools and toolbars for creating documents and typing mathematical formulae in a user–friendly way.

The contents of the first file (Figure 1 in the appendix) describes the basic notions of the tuple theory. It gives the definitions of the tuple concept, the definitions of operations that can be performed on tuples (reversion, concatenation, etc), and propositions involving the defined concepts. The second file (Figure 2 in the appendix) stores axioms and definitions in the domain of natural numbers, where the natural numbers are expressed by the Peano axioms.

Note that this is only a small example. In practice we may have dozens of files, each with hundreds of formulae. Case studies that involve files with a large number of formulae are, for example, the algorithm synthesis [11] and the Gröbner rings [8] case studies. In the near future, a didactic project using *Theorema*, called CreaComp, will also produce large knowledge bases of formulae.

Let us now assume that we want to investigate some of the properties of the length of tuples. For such a case study we need to use basic knowledge about tuples and natural numbers. Since the two files already contain the formalization of this knowledge, we would like to use this knowledge, combine it, extend it, and produce a third file containing the new knowledge (Figure 5). Furthermore, let us assume that we want to select some of the knowledge contained in some of the files and give it as an input to one of the automated reasoning tools of *Theorema*.

For doing this, we need means to access the desired knowledge within the given files. The most natural way to access formulae and groups of formulae is access by position in files, by section and subsection headings, keywords or labels. Since this is not yet possible in the current, native Mathematica notebooks, we need tools for transforming hierarchical section headings, keywords and individual labels of formulae into unique composite labels, which is one of the main objectives of the paper.

3 Description of the Label Management Tools

3.1 Starting Point for the Development of the Tools

The implementation of our tools is done in the frame of Mathematica [37] and *Theorema* [6, 12] which is built on top of Mathematica. As a mathematical editing environment, Mathematica offers a very good front end that gives the possibility of combining text, mathematical expressions, graphics, and code in the same document, called notebook. Another feature of Mathematica, which is used by *Theorema*, is two–dimensional syntax programming (see `MakeExpression` in [37], Section 2.9.17).

The tools for label management described in this paper, take as input Mathematica notebooks which contain comments in text cells and predicate logic formulae in input cells. The formulae are given in *Theorema* external syntax. Also, in these notebooks, labels of various kinds (section heading, key words like "definition", "theorem" etc., and individual labels) can be attached to formulae and groups of formulae. For this, we developed a particular Mathematica stylesheet. A Mathematica stylesheet is a special kind of notebook that defines the styles to be used in other notebooks ([37] Section 2.10). By our stylesheet keywords like "definition", "theorem", "property", etc. can be attached to entire sections, subsections, etc. Formulae that occur under these headings will have attached the keyword given by the style of the headings. If desired, the user can also override these keywords by keywords at the level of individual formulae. With the help of this stylesheet, the label management tools can identify, select, and re–combine formal parts of documents and use them, for example, as input to automated reasoners.

The documents processed by the label management tools operate on libraries of *Theorema* notebooks. A library is a collection of Mathematica notebooks using the above stylesheet. By the tools described below it is guaranteed that the notebook labels are unique. Also, we provide an extra index file that lists all notebooks in the library and also describes the mutual inclusion of notebooks in the library, see below. (In fact, in addition, the Mathematica package facility is used in the organization of the notebooks in the library for speeding up re–loading of notebooks. However, since this is only a technical detail which does not add to the organizational functionality of our tools, we do not refer to packages anymore in this paper).

In the following, we will describe the main tools which we designed and implemented for achieving the objectives specified in Section 2. For the convenience of the user these tools can be accessed also by a new *Theorema* toolbar, called 'Library Utilities' which, however, we do not describe in this paper.

3.2 Tool for Systematic Generation of Hierarchical Labels

An example of a *Theorema* notebook is given in Figure 1 of the appendix. It has a notebook title ("Basic Notions: Tuples") and a notebook label ("BN:Tuples"). Formulae, in input cells, are grouped under section and subsection headings.

Hierarchically grouping cells in sections, subsections and so on, is a feature of Mathematica notebooks. It is common practice that larger notebooks have chapters, sections and so on, each represented by groups of cells. The extent of these groups is indicated by a bracket on the right ([37]), which can also be conveniently used for optical contraction of sections to obtain an easy overview.

The headings of the cell groups, the notebook title, the notebook label, and labels of individual formulae are the components used for generating and attaching unique composite labels to all formulae and groups of formulae in the notebook. If the notebook label is not present in the document, a notebook label is generated from the notebook title such that every notebook in the notebook library has a unique identifier. For this reason, the notebook title is a mandatory element in the *Theorema* notebooks. User given notebook labels are checked against the list of existing notebook labels (extracted from the library index file). The user is notified when the notebook label is already used by another *Theorema* notebook.

From the notebook title, notebook label, section headings, etc. provided by the user our tool automatically generates composite labels for each section, subsection, etc., and individual formula in the notebook. These composite labels are generated in three variants which we call long, short, and decimal composite labels, respectively.

In Figure 1, for example the user provided notebook and theory labels. The user also provided all of the headings in the notebook, among them "Relation on Tuples: Element". The generated labels are: "BN:Tuples.Propositions Involving the Definitions Above.Relations on Tuples: Element" for the long label variant, "BN:Tuples.ProInvDefAbo.RelTupEle" for the short label variant, and "BN:Tuples.5.3" for the decimal label variant.

The short variant of the label is obtained from the long variant by a simple string truncation algorithm, with some proviso for preserving uniqueness. The period in the above labels variants plays the role of a separator, displaying the composite structure of the labels generated.

In Mathematica, notebooks are represented as Mathematica expressions. The label generating routine takes as input the Mathematica expression of a *Theorema* notebook. The notebook expression has a recursive structure, reflecting the grouping and sub–grouping of cells in the notebook. Correspondingly the label generating routine proceeds recursively. The label variants are created by concatenation operations, where the operands are, depending on the label variant, the text of the heading, the short text of the headings (obtained by string truncation algorithms), and notebook counters. Eventual user–given labels are taken into account for the generation of the short and long label variants. The notebook label is prepended to each of the labels so that any resulting label we look at, in the *Theorema* notebook, contains the notebook label as a substring. Because the notebook label is unique among notebook labels in the library, we are sure that the generated labels uniquely identify the formulae and groups of formulae within the library.

Figures 3 and 4 in the appendix show parts of the notebooks in Figures 1 and 2 after they have been processed by the label generating routine. The labels that can be seen right above the formulae and groups of formulae are the decimal label variants. The short label variants are not shown. From now on, when we use the word 'label' we refer to one of the three variants of a label.

Now, groups of formulae and individual formulae of *Theorema* notebooks can be referenced by composite labels and can be used for composing new *Theorema* notebooks and knowledge bases as an input to formal reasoners as described in the next two sections (see Figure 5 in the appendix).

3.3 Tool for Including Formal Parts of Notebooks into Other Notebooks

When we write a new *Theorema* notebook we may now also include parts of already existing *Theorema* notebooks in the library. For this we implemented an '`Include`' command which takes all formulae referenced in its arguments and copies them, together with their unique labels, into the current notebook. This gives us the possibility to concentrate knowledge dispersed in various notebooks in the library in one *Theorema* notebook and use this new notebook independently of the library. Of course, there is also a possibility to list the new notebook in the library index of the current library or some other library. This is particularly convenient when distributing libraries over the web.

The '`Include`' command has the structure

$$\texttt{Include}[\texttt{Label}_1, \texttt{Label}_2, \ \ldots\ , \texttt{Label}_n, \ \texttt{Option}]$$

whose functionality should self–explanatory: Take the collections of formulae referenced by the composite labels $\texttt{Label}_1, \ \ldots\ , \texttt{Label}_n$ from the current library and copy them into a new version of the notebook that contains the '`Include`' command and cancelled the '`Include`' command.

There are two settings of the '`Options`' argument of the '`Include`' command. With the first setting, the original composite labels of the formulae included are kept unchanged. With the second setting, the composite label of the '`Include`' command will be prepended to the labels of the formulae included. If no setting for '`Option`' is given, the latter described setting is considered. (Figure 5 in the appendix shows a *Theorema* notebook before applying this tool to it.)

3.4 Tool for Using Selected Formal Parts of Notebooks

The various reasoners (provers, simplifiers, and solvers) of *Theorema* can be called by instructions of the following structure

$$\texttt{Reason}[\texttt{Goal}, \texttt{using} \rightarrow \texttt{KnowledgeBase}, \texttt{by} \ \rightarrow \ \texttt{ReasoningMethod}],$$

where 'Reason' can be '`Prove`', '`Compute`', '`Solve`'; '`KnowledgeBase`' is expressed by

$$\langle \texttt{Label}_1, \ \ldots\ , \texttt{Label}_n \rangle$$

and $\texttt{Label}_1$, ... , $\texttt{Label}_n$ are composite labels of collections of formulae in the notebook library.

For example

$$\texttt{Prove}[\text{ “LenTpl.3.1”},\ \texttt{using} \rightarrow \langle \text{“}\texttt{BN : Tuples}\text{”}, \text{“}\texttt{NN : Basic}\text{”}\rangle, \texttt{by} \rightarrow \texttt{TupleEqIndProver}]$$

where ‘TupleEqIndProver’ is a *Theorema* prover that combines rewriting and induction over tuples.

Alternatively, we also implemented the following ‘Theory’ construct:

$$\texttt{Theory}[\texttt{Label}, \langle \texttt{Label}_1, \ \ldots\ , \texttt{Label}_n \rangle]$$

which is something like a temporary assignment of a new label to specified collections of formulae so that we can formulate calls to reasoners also in the following form

$$\texttt{Reason}[\texttt{Goal}, \texttt{using} \rightarrow \texttt{Label}, \texttt{by} \rightarrow \texttt{ReasoningMethod}].$$

For example, the above call can also be formulated in the following way:

$$\begin{array}{l} \texttt{Theory}\ [\text{“}\texttt{Tuples and Natural Numbers}\text{”}, \langle \text{“}\texttt{BN : Tuples}\text{”},\ \text{“}\texttt{NN : Basic}\text{”}\rangle]; \\ \texttt{Prove}\ \ [\text{“}\texttt{LenTpl.3.1}\text{”}, \\ \qquad \texttt{using} \rightarrow \text{“}\texttt{Tuples and Natural Numbers}\text{”},\ \texttt{by} \rightarrow \texttt{TupleEqIndProver}]. \end{array}$$

4 Conclusion

We designed and implemented simple tools that allow to reference specific parts of collections of mathematical knowledge bases organized in libraries of *Theorema* notebooks. These tools generate systematically composite hierarchical labels for all the sections, subsections, etc. and the individual formulae of *Theorema* notebooks from section headings and individual labels of formulae in the original notebooks provided by the user. With these tools one then can quickly compose specific new notebooks and knowledge bases as input to the formal reasoners of the *Theorema* system using the composite hierarchical labels. These tools can also be used for knowledge bases in other systems by translation of the formulae formats between systems.

Seemingly, label management is a trivial part of mathematical knowledge management. However, we believe that, in fact, systematic and efficient label management is quite significant for the user–friendliness of future mathematical knowledge management systems and needs systematic treatment. For different purposes within MKM the choice of labels must meet different criteria. For example, for human readers of mathematical knowledge bases (e.g. in the form of *Theorema* notebooks) long textual labels may be preferable whereas in the presentation of proofs short version of labels are desirable in order not to disrupt the flow of the proof presentation. In the extreme case, if the proof presentation style of ‘Focus Windows’ is used (see [34]), one even does not need labels in proof

presentations. However, at the same time, labels as references for organizing new knowledge bases from given ones, for example as input to provers, are very important. Thus, flexible label management tools on top of the logic tools of MKM systems are necessary.

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. P. B. Andrews, M. Bishop, C. E. Brown. System Description: TPS: A Theorem Proving System for Type Theory. In D. McAllester (Ed.): *Automated Deduction - CADE-17; 17th International Conference on Automated Deduction*, Pittsburgh, PA, *Lecture Notes in Artificial Intelligence* Vol. 1831, Springer–Verlag, 2000, pp. 164 – 169. Proceedings of CADE – 17.
3. A. Asperti, B. Buchberger, J.H. Davenport, James Harold (Eds.): *Proceedings of the Second International Conference, MKM 2003 Bertinoro, Italy, February 16–18, 2003* Lecture Notes in Computer Science Series, Vol. 2594, 2003, X, 225 p. Also available online. Softcover ISBN: 3–540–00568–4.
4. C. Benzmller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, A. Meier, E. Melis, W. Schaarschmidt, J. Siekmann, V. Sorge. OMEGA: Towards a Mathematical Assistant. In W. McCune (Ed.): *Automated Deduction - CADE 14, Lecture Notes in Artificial Intelligence*, Vol. 1249, Townsville, North Queensland, Australia, July 1997. Springer–Verlag.
5. B. Buchberger. Mathematical Knowledge Management Using *Theorema*. In [10]
6. B. Buchberger. *Theorema*: A short introduction. *The Mathematica Journal*, 8(2):247–252, 2001.
7. 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 1–4, 2003), Mirton Publishing, ISBN 973–661–104–3, pp. 2–26.
8. B. Buchberger. Gröbner 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.
9. B. Buchberger. Computer–Supported Mathematical Theory Exploration: Schemes, Failing Proof Analysis, and Metaprogramming. To be published in Proceedings of Artificial Intelligence and Symbolic Computation AISC 2004, RISC, A–4232 Schloss Hagenberg, September 22–24, 2004.
10. B. Buchberger, O. Caprotti (Eds.): *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.
11. B. Buchberger, A. Craciun. Algorithm Synthesis by Lazy Thinking: Examples and Implementation in *Theorema*. In F. Kamareddine, (Ed.): *Proc. of the Mathematical Knowledge Management Workshop, Edinburgh, Nov. 25, 2003, Electronic Notes in Theoretical Computer Science*, Vol. 93, 18 Feb. 2004, pp. 24–59, www.elsevier.com/locate/entcs. ISBN 044451290X.
12. B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger. The *Theorema* Project: A Progress Report. In M. Kerber and M. Kohlhase (Eds.): *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. A.K. Peters, Natick, Massachusetts, pp. 98–113. ISBN 1–56881–145–4.
13. B. Buchberger, G. Gonnet, M. Hazewinkel (Eds.): *Annals of Mathematics and Artificial Intelligence*, Volume 38, Number 1–3, May 2003. Kluwer Academic Publishers, ISSN 1012–2443.
14. O. Caprotti, D. Carlisle. OpenMath and MathML: Semantic Mark Up for Mathematics. In *ACM Crossroads*, ACM Press, 1999.
15. K. Chan and D. Yeung. Mathematical Expression Recognition: A Survey, *International Journal on Document Analysis and Recognition*, Vol. 3, No.1, pp. 3–15, August, 2000. Springer–Verlag Heidelberg ISSN: 1433–2833 (Paper) 1433–2825 (Online).
16. R. L. Constable, S. F. Allen, H.M.Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P.Mendler, P. Panangadenden, J. T. Sasaki, S. F.Smith. *Implementing Mathematics with the Nuprl Development System*, Prentice–Hall, Enlgewood Cliffs, NJ, 1986.
17. The Coq proof assistant. (http://coq.inria.fr/coq-eng.html)
18. 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, Edinburgh, Scotland.
19. M. J. C. Gordon, T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Reasoning.* Cambridge University Press, 1993.
20. Helm: Hypertextual Electronic Library of Mathematics. (http://helm.cs.unibo.it/)
21. 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/)
22. Isabelle. Developed at Cambridge University (Larry Paulson) and TU Munich (Tobias Nipkow). (http://www.cl.cam.ac.uk/Research/HVG/Isabelle/index.html)
23. jEditOQMath Tools Package. http://www.activemath.org/projects/jEditOQMath/
24. JOME: Java OpenMath Editor. Written by Laurent Dirat. (http://mainline.essi.fr/wiki/bin/view/Jome/WebHome).
25. A. Kohlhase. CPoint Documentation. (http://aiki.ccaps.cs.cmu.edu/DownloadIndex.html#CPoint)
26. M. Kohlhase. OMDoc: An Infrastructure for OpenMath Content Dictionary Information. In *ACM SIGSAM Bulletin*, volume 34, number 2, pages 43–48, 2000.
27. 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.
28. T. Kutsia, K. Nakagawa. An Interface between Theorema and External Automated Deduction Systems. In S. Linton and R. Sebastiani (Eds.): *Proceedings of 9th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning*, June 21–23, 2001, pp. 178–182, Siena, Italy.
29. P. Libbrecht, E. Melis, C. Ullrich The ActiveMath Learning Environment: System Description. In Calculemus Workshop at IJCAR, pages 173–177, 2001.
30. D.W.Lozier. NIST Digital Library of Mathematical Functions. In [13] pp.105–119.
31. The Mizar System. Developed at the University of Warsaw, directed by A. Trybulec. (http://mizar.uwb.edu.pl/system/)
32. S. Owre, J. Rushby, N. Shankar, D. Stringer–Calvert. PVS: An Experience Report, In D. Hutter, W. Stephan, P. Traverso, M. Ullman (Eds.): *Applied Formal Methods - FM–Trends 98.* LNCS vol. 1641, pp. 338–345. Springer–Verlag, Germany.

33. F. Piroi. Tools for Using Automated Provers in Mathematical Theory Exploration. Ongoing PhD thesis, to be finished in Autumn 2004.
34. F. Piroi, B. Buchberger. Focus Windows: A New Technique for Proof Presentation. In J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, V. Sorge, (Eds.): *Artificial Intelligence, Automated Reasoning and Symbolic Computation. Proceedings of Joint AISC'2002 - Calculemus'2002 Conference*, July 1–5, 2002, Marseille, France. Volume 2385 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, 2002, 290–304. (ISBN 3–540–43865–3.)
35. S. Rockey. Mathematics Digitization, at Cornell University, Mathematics Library. (http://www.library.cornell.edu/math/digitalization.php)
36. W3C Math Home: What is MathML? (http://www.w3.org/Math/)
37. S. Wolfram. *The Mathematica Book*, Fifth Edition. Wolfram Media Inc., 2003.

Appendix

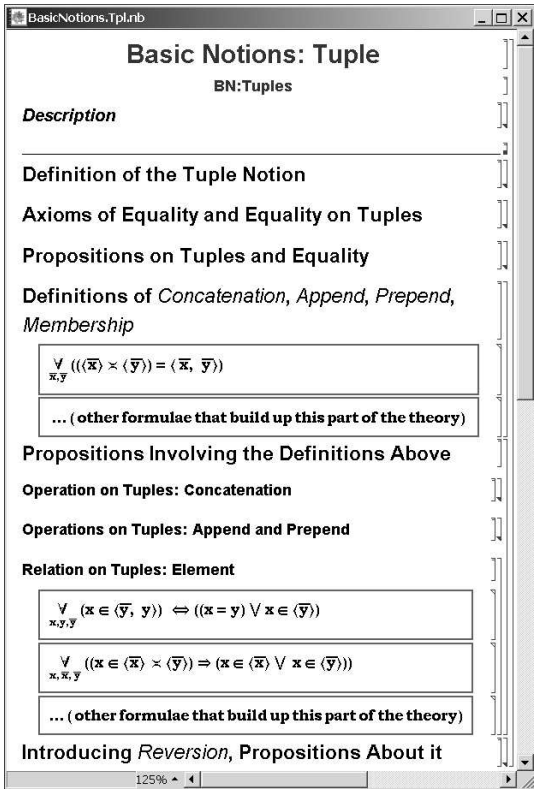

Figure 1.

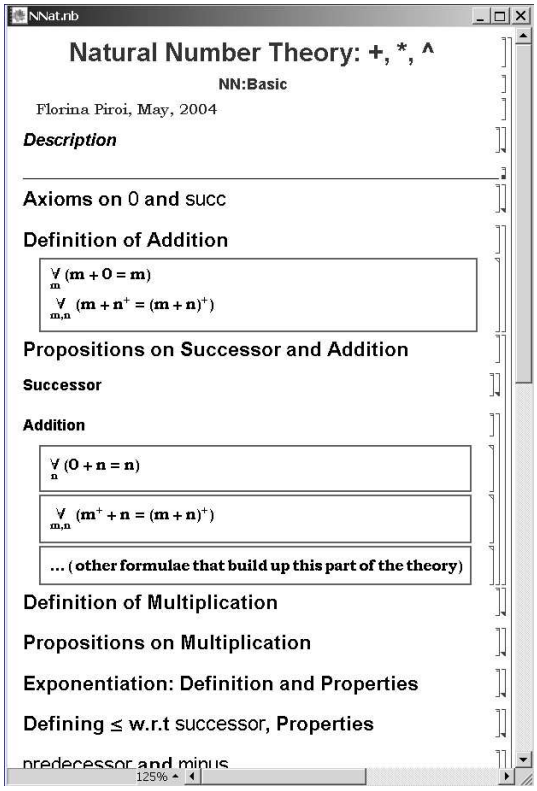

Figure 2.

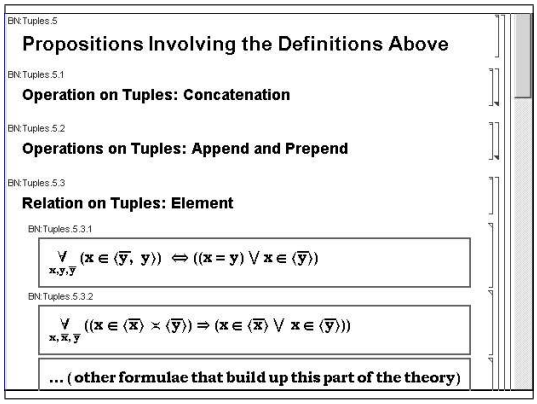

Figure 3.

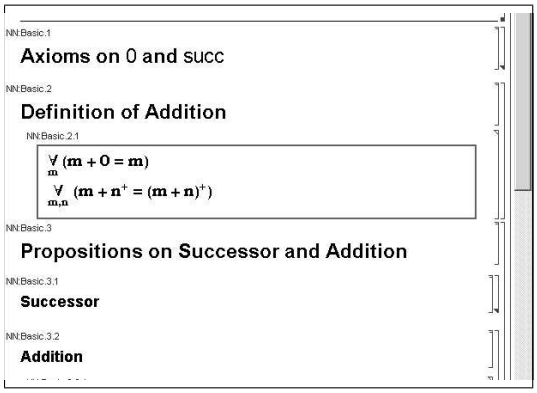

Figure 4.

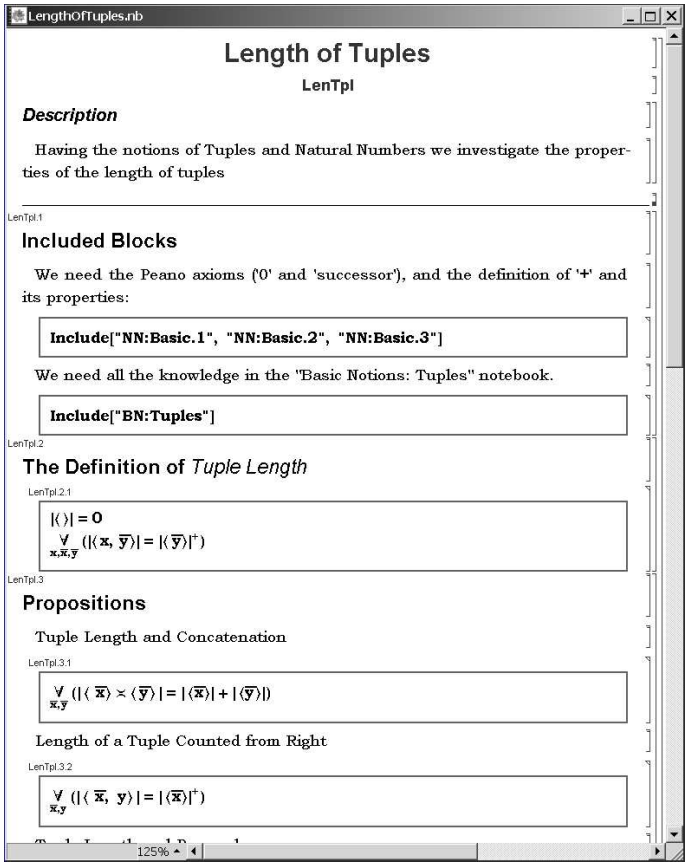
LengthOfTuples.nb
Length of Tuples
LenTpl
Description
Having the notions of Tuples and Natural Numbers we investigate the properties of the length of tuples
LenTpl.1
Included Blocks
We need the Peano axioms ('0' and 'successor'), and the definition of '+' and its properties:
Include["NN:Basic.1", "NN:Basic.2", "NN:Basic.3"]
We need all the knowledge in the "Basic Notions: Tuples" notebook.
Include["BN:Tuples"]
LenTpl.2
The Definition of Tuple Length
LenTpl.2.1
|⟨⟩| = 0
LenTpl.3
Propositions
Tuple Length and Concatenation
LenTpl.3.1
Length of a Tuple Counted from Right
LenTpl.3.2
125%

Figure 5.