Adapting Mainstream Editors for Semantic Authoring of Mathematics

G.Goguadze, A.González Palomo

Table of contents

1 Introduction 1

2 Defining OMDoc XML-filter for OpenOffice 1

2.1 Annotating the Document Structure 2
2.2 Editing Mathematical Formulas 3
2.3 Restrictions of OpenOffice Format and Interface 3

3 Customizing TeXmacs for Authoring OMDoc 4

4 Conclusion and Future Work 5

Bibliography 5

1 Introduction

Annotating mathematical knowledge with semantics has obviously become of a major importance for systems dealing with mathematical knowledge management. We investigate various possibilities for authors to produce semantically annotated mathematics using existing general purpose editors such as OpenOffice[2] and TeXmacs[3].

We use the semantic markup language OMDoc[1] for the representation of mathematical knowledge. OMDoc is an XML language which is an extension of an OpenMath[1] standard for representation of mathematical formulas. It classifies the items of mathematical knowledge in definitions, axioms, theorems, lemmas, corrollaries, proofs and other types of conceptual markup needed for building the ontology of mathematical knowledge.

Annotating mathematical documents with semantics is a tedious process, since authors have to provide an amount of additional annotations to their documents. We have been investigating different styles of authoring tools that can help authors not familiar with XML languages, since for those editing the XML document directly is a non trivial and expensive process.

We consider two different styles of WYSIWYG editors: OpenOffice[2] which is the open source version of an advanced office program able to import and export Microsoft Office documents and TeXmacs[3], macro-based editor combining some features of Emacs[4] and LaTeX[5]. Each of these editors have their pros and cons in usability for annotating the documents with semantics and their architectural particularities impose different technical approaches and restrictions.

2 Defining OMDoc XML-filter for OpenOffice

The OpenOffice suite is a mainstream editor - an open source alternative to the Microsoft® Office suite.

It can import and export any Microsoft® Office document. Due to the fact that the internal representation format of an OpenOffice document is XML, it is relatively easy to convert it via XSL transformations to other document formats. The latest version of the editor provides a possibility to plug the external XSL stylesheets that convert OpenOffice documents to any needed format. There already exist converters from OpenOffice to LaTeX, DocBook[6] and other popular formats.

1. See http://www.openmath.org
2. See http://www.openoffice.org
3. See http://www.texmacs.org
4. Simple and powerfull open source text editor, see http://www.gnu.org/software/emacs/emacs.html
5. A document typesetting system, see http://www.latex-project.org
6. A general purpose XML and SGML document type, see http://www.docbook.org

The first step towards converting the OpenOffice document to OMDoc is simply mapping the document structure without introducing additional annotations to the document.

This has been realized via a simple XSL stylesheet that produces OMDoc document, consisting of simple text elements not carrying any mathematical semantics. The way how the document structure is transformed and which pieces of text belong together depends on the original formatting of the OpenOffice document. For instance, one would expect that when having a PowerPoint® presentation each text box is represented as a separate text item and the lists on a slide are represented as lists inside a text element.

Note that a single XSL stylesheet can handle the convertion of any Microsoft Office document: Microsoft® Word document, PowerPoint® Presentation or an Excel Spreadsheet, since the XML representation format of OpenOffice is using the same grammar for all of them.

2.1 Annotating the Document Structure

The next step towards semantic representation of the knowledge in the document is splitting the text in parts and annotating them with semantics. We need to introduce types of mathematical items such as definitions, theorems etc., relations between these items and other meta information. For this we define an OMDoc template, which the user has to load when creating a new document. The template defines custom styles for frames and paragraphs which are the building stones for macro- and micro- structure of the annotated document.

The items of OMDoc and their metadata are represented by frames and for classifying the content of each item the paragraph styles are used. Authors can assign identifiers to the knowledge items, provide links between items in the same document or just copy and paste hyperlinks from the browser, insert tables and animations, etc..

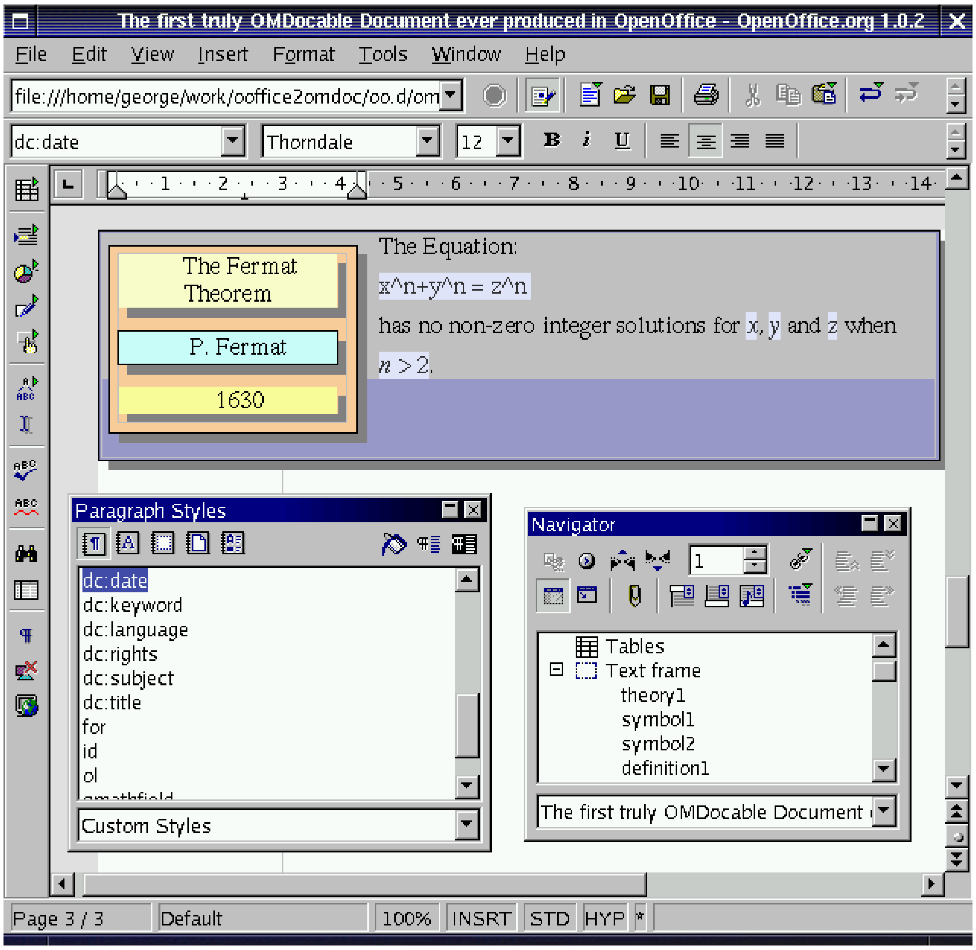

Figure 1. A snapshot of an OpenOffice session using an OMDoc template.

As one can see from the Figure above, the author can use the OpenOffice style palettes to select the OMDoc styles he wishes to use, or navigate through the document using the OpenOffice navigator window.

2.2 Editing Mathematical Formulas

The procedure used for marking up the document structure, selecting parts and associating styles with them, does not work well for formulas because of their high structure density, so we need to automate part of the structure extraction using a parser.

The one we have readily available that can produce OpenMath is QMath[7], which accepts custom symbols. Also we have just made parsers for internal use in ActiveMath[8] that translate Maple™[9] or Mathematica™[10] syntax into OpenMath, which could be used for this purpose.

Our current scheme consists on marking the mathematical expressions with a custom style called "qmathfield". First, the user marks the expression with this style, and after exporting to XML, we feed it into a custom[11] version of QMath that outputs the rest of the document verbatim, while parsing that fragments to obtain the OpenMath encoding and insert it in the output.

The symbols in QMath are grouped in files called "contexts". The idea is that the author can specify what context files are to be loaded in order to have the symbols he needs for a given document. The standard symbols defined are those in the OpenMath content dictionaries.

However, users expect to be able to just write the expressions without having to know what context files they need. QMath allows to load all the contexts at once, but when two or more text tokens have different translations, only the last one is retained. This causes the masking of some symbols.

To solve that, we are writing a special OpenOffice context file that contains all the symbols whithout overlapping, by assigning the shorter tokens to the most usual symbols, and then having longer alternatives for the rest. This allows us to prepare a single sheet with all the symbols available to OpenOffice authors, which they can use to look them up without more consideration.

The author can also annotate each symbol he enters with semantics, different from the default one. For this, he must select the symbol, change to the character style menu and select the symbol he needs in the list of all symbol from OpenMath content dictionaries. In this case, however, he can use the symbol only in the prefix form, applied to the arguments in the brackets, since this information is not provided to QMath. Alternatively, we could assign the default fixity values to the OpenMath symbols listed.

2.3 Restrictions of OpenOffice Format and Interface

When trying to customize the interface of OpenOffice, one can see that it is not really designed for semantic annotations, the style pallets that we use were designed particularly to facilitate the customization of presentation styles of the document.

In order to provide more flexible pallets the OpenOffice interface has to be extended by providing BASIC macros or making a Java plugin. OpenOffice has support for both of these languages.

7. QMath is a simple customizable parser for mathematical formulas, that produces OpenMath expressions. It suggests a CAS-like linear syntax, see http://www.matracas.org/qmath

8. ActiveMath is an Interactive Learning Environment for Mathematics, using OMDoc for encoding the course materials, see http://www.activemath.org

9. A Computer Algebra System, see http://www.maplesoft.com/products/maple

10. A Computer Algebra System, see http://www.wolfram.com/products/mathematica

11. This is similar to previous versions QMath that produced MathML instead of OpenMath and allowed to write HTML constructs that were output verbatim.

The OPENOFFICE XML-filter provides a possibility to validate the produced XML document, but it is not possible to backtrack the errors and highlight the places in the original document where the errors were made. The validation and additional interfacing could be realized by implementing the corresponding Java component as well as by writing BASIC macros.

Each of these possibilities is already some external programming. As opposed to it, all we did so far was mostly a matter of customizing the interface for our needs and defining XSL templates which still brought us surprisingly far. This shows the usefulness of XML being an internal representation as opposed to the internal representation of, for instance, Microsoft® Word format.

3 Customizing TEXMACS for Authoring OMDOC

In contrast with OPENOFFICE, TEXMACS is a structured editor with specific features for editing mathematical documents, which makes it a much better platform for development of authoring tools in this area.

During the last month, the authors assisted Joris van der Hoeven in improving TEXMACS for producing XML output that could be processed with our XSLTs to produce OMDOC out of it.

The way to construct OMDOC-specific elements is to augment the existing ones with the additional data they need. This is done by defining TEXMACS macros that take the precise number of arguments and translate them from TEXMACS XML into OMDOC with XSLT. For example, a "Theorem" item in TEXMACS takes just one argument, the text body. The custom OMDOC "Theorem" macro takes three arguments: id, metadata, and body. Then, the OMDOC mode provides a macro, available through a menu entry called "Buffer as OMDOC", that converts all the standard items into the custom OMDOC equivalents, with empty placeholders for the extra parameters, as shown on the Figure below.

So it is possible to load a LATEX document and convert it so that after filling in that extra data, the result is an OMDOC document.

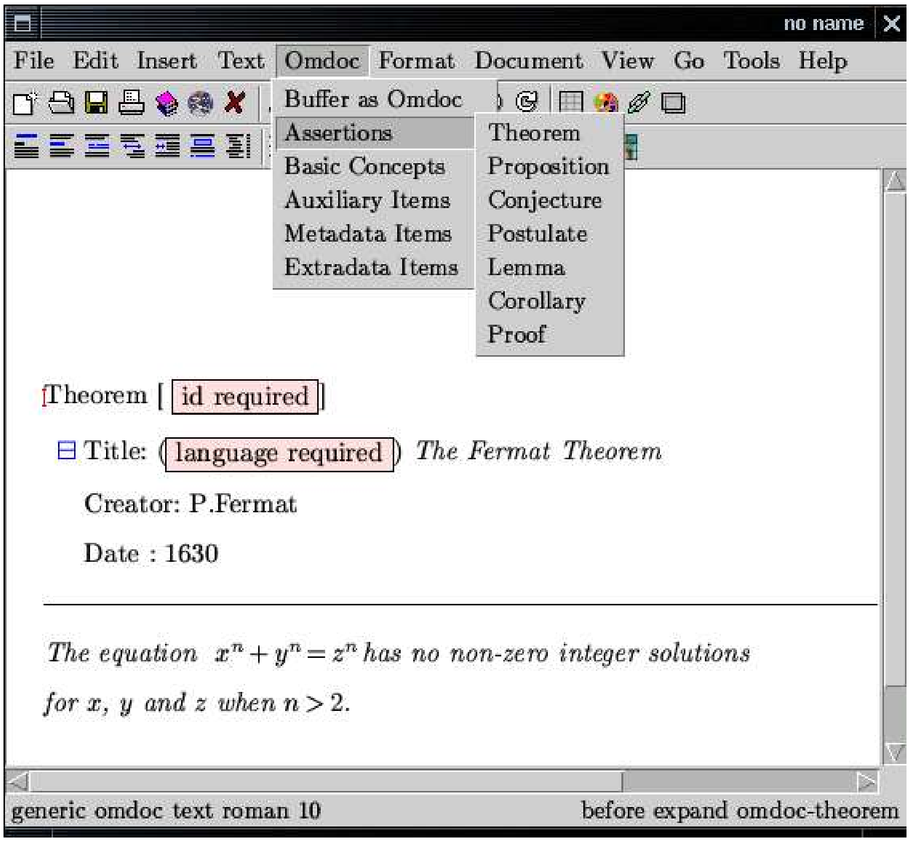

Figure 2.

Formulas in TeXmacs are just concerned with presentation, but the high degree of structure in them (similar to TeX or MathML) makes we think that in the long term it will provide better results.

Similarly to the case of OpenOffice, one could use QMath for transforming formulas. TeXmacs allows for even more flexibility for using QMath, since one could easily introduce menu entries for importing mathematical contexts to be used by QMath, and macros could be used to annotate the symbols used in formulas with proper semantics, naming the corresponding content dictionary of OpenMath selecting it from the menu. The stylesheet, transforming the XML format exported by TeXmacs will convert the formulas into QMath syntax, generating the corresponding symbol declarations that QMath would use for further translation to OpenMath.

4 Conclusion and Future Work

Summarizing the work that has been done so far, we have shown the possibilities of authoring semantic mathematics using mainstream editors dealing with common formats like Microsoft® Office documents, LaTeX, TeXmacs and investigated the possibilities to customize or extend these editors in order to satisfy our wish to obtain a simple tool for producing documents in OMDoc format.

Possible directions for further work could be :

- Assisting the author by interactive validation of the document structure and semantics.
- Providing better ways to author mathematical formulas including communication with external systems like CAS or theorem provers.
- The export of annotated TeXmacs documents to OMDoc and the import of OMDoc documents back into TeXmacs can be implemented directly as a Scheme[12] plugin, without the necessity to pass through an intermediate XML format.

Finally, the authors want to acknowledge the helpul attitute of Joris van der Hoeven while we collaborated with him in the process of developing the TeXmacs plugin.

Bibliography

[1] M. Kohlhase, OMDoc: Towards an OpenMath Representation of Mathematical Documents, Seki Report, FR Informatik, Universität des Saarlandes, 2000.

[2] The OpenOffice.org Project. Sun Microsystems Foundations of Office Productivity in a Networked Age, 2000

[3] GNU TeXmacs: A free, structured, wysiwyg and technical text editor, Actes du congrés GUTenberg (no. 39-40), Metz, Mai 2001, pages 39-50.

12. A programming language, dialect of Lisp, see http://www.swiss.ai.mit.edu/projects/scheme