Easily Editing and Browsing Complex OpenMath Markup with SWiM

Christoph Lange* Alberto González Palomo†

July 21, 2008

We present how the mathematical semantic wiki SWiM has been enhanced towards support for editing content dictionaries (CDs) in the semantic markup language OpenMath. The ongoing revision of the OpenMath CDs for the 3rd version of the standard has motivated several enhancements to the SWiM user interface: a structural editor for CDs and their most common elements, particularly addressing notation definitions for symbols, the integration of a visual OpenMath formula editor, and a form-based editor for metadata. We show how OpenMath CDs imported from the filesystem or a Subversion repository are split into handy fragments that are convenient to edit and navigate, and reassembled on export.

1 Introduction and Motivation

OpenMath [3] is a widely used semantic markup language for mathematical formulæ and for content dictionaries (CD) that semi-formally define new mathematical symbols – their informal description, their formal description in terms of other symbols, and optionally a declaration of their type and their human-readable notation. Editors for OpenMath formulæ exist, and the CDs approved by the OpenMath Society are presented for convenient reading and browsing on the web [22], but to date there has not been an editor for CDs. Creating custom CDs and symbol definitions is a common use case OpenMath has been designed for [5]: Authors need to do it when the mathematical concepts they are concerned about have not been covered by the existing CDs sufficiently deeply or rigorously [5, 1].

Editing support for CDs does not only require assistance with writing and structuring, but also with browsing and collaborating. For example, CDs frequently reuse symbols from other CDs, which makes an easy navigation from the occurrence of a symbol in a formula to the place of its definition desirable. For collaborating on CDs there is currently no other support than keeping them as files in a shared repository. This particularly makes it hard to manage notational changes. For one mathematical symbol, there can be multiple notations, and they can change[1]. Here, we mainly consider the case of having one "standard" notation per symbol, which is subject to fine-tuning; assume e. g. that the OpenMath Society decide that the default rendering of the multiplication operator should no longer be the "invisible times" symbol, but $\times$. Verifying such a change is easiest by looking at occurrences of that symbol in realistic formulæ. In the traditional workflow, this requires finding all CD files and other documents that use the symbol whose notation has been changed using text or

*Computer Science, Jacobs University Bremen, ch.lange@jacobs-university.de
†Computer Science, Saarland University, alberto.gonzalez@matracas.org
[1]For general background, we refer to [19] and [14].

XML search, regenerating the XHTML+MathML versions of these documents with an XSLT processor [19] – possibly automated by a makefile –, and then opening them in a browser, scrolling to the sections containing formulæ.

Below we present how the semantic wiki SWiM facilitates this workflow: how the knowledge obtained from OpenMath CDs is represented and used to support navigation, how CDs and formulæ are edited, and how the system reflects changes to notation definitions.

2 SWiM, a Semantic Wiki for Mathematics

Wikis are well-established for web collaboration, so we aim at supporting collaboration on OpenMath CDs by adapting SWiM [15] accordingly. SWiM already supported the OMDoc semantic markup language [12] with OpenMath formulæ; now we have added support for CDs and for notation definitions in the pattern-based language proposed for MathML 3 [14], which is likely to be adopted for OpenMath 3. The *mmlproc* renderer [14] renders formulæ according to the notations defined for the symbols they use. Mathematical documents can be edited in SWiM, but also imported and exported and thus exchanged with external repositories and edited with legacy editors.

3 Knowledge Representation and Navigation

SWiM exploits the structural semantics of CDs to support editing and browsing. An OpenMath 2 CD consists of metadata (e. g. the review date) and symbol definitions (at least of a symbol name and a mandatory and authoritative informal, natural-language description). Optionally a symbol definition can have metadata, specifications of mathematical properties – informal (so-called "commented mathematical properties", CMPs) or formal ones (FMPs) –, and examples. FMPs and examples contain formulæ, which in turn use symbols, possibly from other CDs [3]. Type signatures and notation definitions for symbols can be given in separate dictionary files.

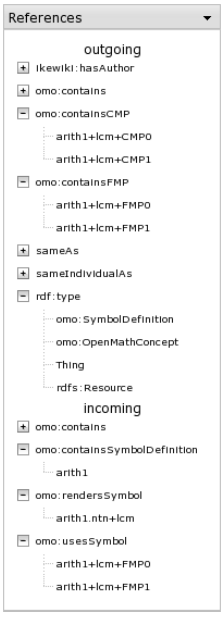
Being a *semantic wiki*[2], SWiM represents knowledge in the graph-like RDF data model that is common on the semantic web. To get most out of the previously listed structural properties of CDs, SWiM extracts them from the OpenMath XML documents to an RDF representation [16]. Nodes and edges in an RDF graph (= wiki pages and links) are labeled with types from vocabularies also called *ontologies*. For the RDF extracted from OpenMath we thus had to design an ontology that models the structural properties of CDs. This ontology reflects the syntactical structures of CDs, as given by the XML schema of the CD language[3], and additionally it models their *semantics* [16]. This semantics is given in the OpenMath specification [3], which is not machine-comprehensible. Consider for example the `CDUses` element that refers to a list of CDs whose symbols are used by the current one. The XML schema can only restrict it to a list of strings that would be allowed for naming CDs, but it does not convey the semantics that any such string is a by-name pointer to an actual CD[4]. The OpenMath ontology models classes and properties for all structural entities found in OpenMath's CD groups, CDs, type signatures, and notation definitions. Properties from common ontologies like Dublin Core were reused where appropriate[5].

[2]For more background on this, we refer the reader to a previous MathUI publication about SWiM [15] and to the proceedings of the 3rd Semantic Wiki Workshop [18].

[3]We use "XML schema" as the generic term for an XML grammar language. Actually this is RELAX NG in the case of OpenMath.

[4]Note that by adding a rule to the ontology it is also possible to *compute* the CDs used by some CD by looking up the CDs of the symbols occurring in its FMPs and examples.

[5]The idiosyncratic metadata vocabulary of OpenMath 2 is likely to be replaced by Dublin Core (DC) in



Wikis facilitate browsing and collaboration on knowledge by encouraging a densely linked network of small knowledge items. Small wiki pages reduce the potential of editing conflicts and allow for giving more focused search results. SWiM exploits typed links on the level of whole pages, displaying them in a navigation box, grouped by type – as shown on the right for a symbol definition. One page can contain a CD, or a subsection of a CD: As subsections of CDs also carry relevant links and annotations and are frequently edited independently from other such units – e. g. when an example for a symbol is edited while leaving an FMP sibling untouched –, we decided to allow wiki pages to represent knowledge on CD level, on symbol level (symbol definitions, type signatures, or notation definitions), and even below symbol level (CMPs, FMPs, examples). When an external CD is imported into SWiM, e. g. from the filesystem or the OpenMath Subversion repository, it is split into these units. The logical containment relationships before splitting are preserved as XInclude [21] links and extracted to RDF, using link types such as *SymbolDefinition–containsExample–Example*. Note, however, that the OpenMath standard considers whole CDs as units that are subject to review or change [3, sect. 4.5], and that for *reading* a CD or a symbol definition it would be inconvenient to manually traverse all links to subsections of interest. Therefore, the XInclude links are resolved automatically when a CD is exported from SWiM or when a CD or a symbol definition is opened for reading, yielding a whole, self-contained CD.

The occurrence of symbols in formulæ is only shallowly represented in the RDF graph, by direct links from FMPs or examples to the symbols they use. Representing the full tree-like functional structure of a formula in RDF would add much overhead and hardly yield additional benefit. If we had a vocabulary for expressing the whole *syntactic* structure of a formula like $\forall x.x \in \mathbb{R} \Rightarrow e^x > 0$ in RDF, as discussed in [20], we would be able to make the reference from the term e^x to the bound variable x explicit (they would point to the same URI), but still we would not be able to express the notion of α-equivalence in the first order logic subsets commonly used for semantic web reasoning (e. g. description logics). Exploiting the full semantics of formulæ should instead be left to a theorem prover, a CAS, or a formula search engine. For supporting editing and navigation, shallow links to symbols already prove useful: for computing dependent CDs (`CDUses`; see above), and for reflecting changes to notation definitions, as we will show in section 4.2. However, the links from individual symbols in rendered formulæ to their definitions can be traversed with the mouse. This linking is achieved by post-processing the parallel markup that *mmlproc* outputs and translating the (cdbase,cd,name) triples to SWiM-internal page URLs.

4 Editing

The SWiM editor is an extension of the visual HTML editor TinyMCE [23]. The structures of the semantic markup are made accessible as special nested HTML tables, which can easily be inserted via tool buttons. Both directions of this conversion are implemented in XSLT. The head line of one such table includes the name of the XML element, e. g. `CDDefinition`, and optionally the list of attributes as "key=value" lines, e. g. the `type` attribute of a signature dictionary, which points to a CD defining the type system used for the signatures. While any desirable markup can be represented like this, it is not user-friendly for deeply nested structures. Therefore, SWiM gives dedicated editing support for certain aspects of the

OpenMath 3. Anticipating this change, we map e. g. the `Name` of a symbol definition to the `dc:identifier` property, and `Description` to `dc:description`.

markup: First, metadata of CDs or symbol definitions are editable via a dedicated form-based metadata editor. Secondly, for some XML elements the tables are arranged more intuitively. For example, notation definitions map a prototype to a rendering (cf. sect. 4.2), which is reflected by the side-by-side arrangement | notation | | prototype | rendering | instead of | notation | prototype | rendering |. Finally, there is a visual formula editor, which we will explain in detail in the following section.

4.1 The Formula Editor

We reuse the visual OpenMath formula editor that originated in the Sentido Mathematical Environment [9] and integrate it as a plugin into TinyMCE in a similar way as done previously into the MathWebSearch formula search engine [13].

Inside TinyMCE, the formulæ are encoded as `span` elements, decorated using CSS. When the cursor is inside one of them, the Sentido toolbar button is highlighted, and the document path displayed below shows the location as "formula", instead of "span".

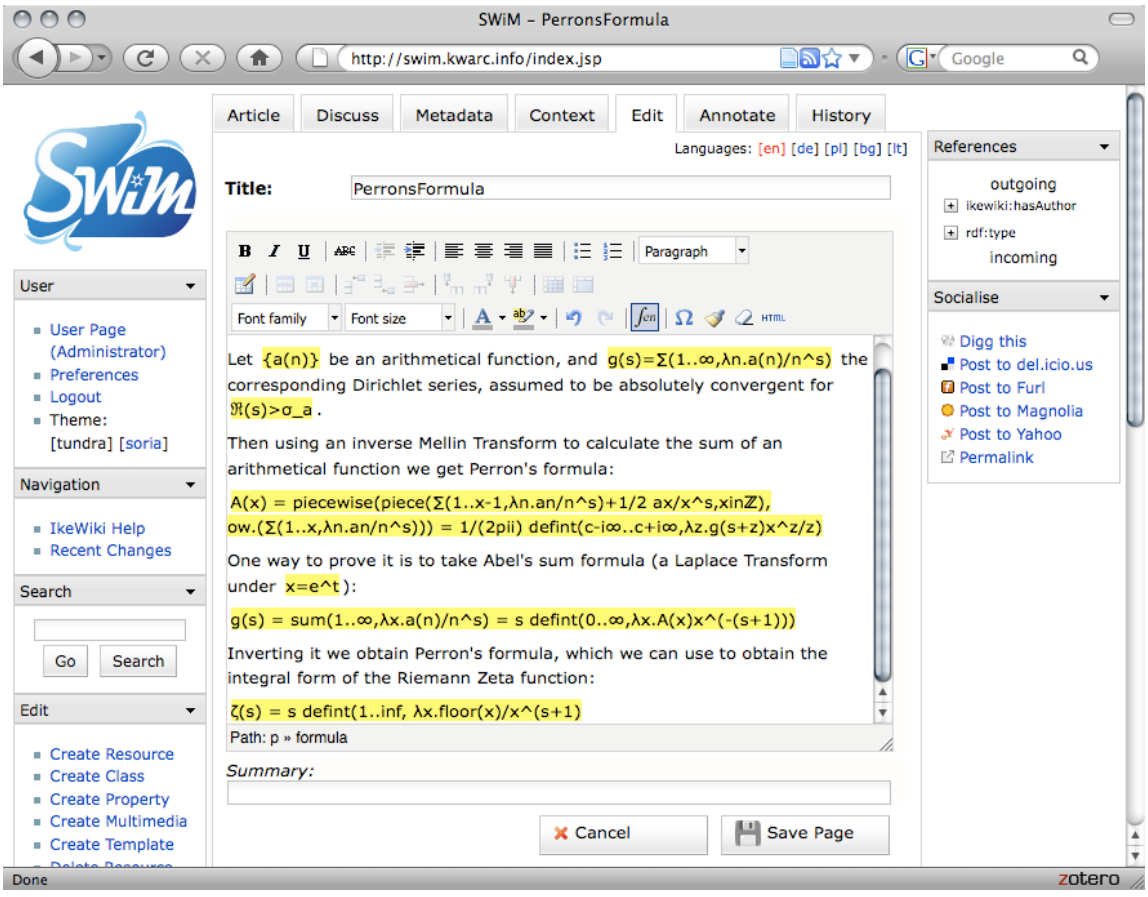


Figure 1: Editing a document in the extended TinyMCE, formulæ marked yellow.

The editor comes as a pop-up window, consisting of an input field for the linear formula, a drop-down menu for selecting the input syntax, a preview area where the 2D formula is shown in Presentation MathML updated in real time as we type in the input field, and a set of collapsible palettes for inserting formula templates. These palettes are XHTML made by hand to include all the symbols from the MathML group of the standard OpenMath CDs, but we plan to make SWiM automatically generate additional ones for other symbols defined in the wiki.

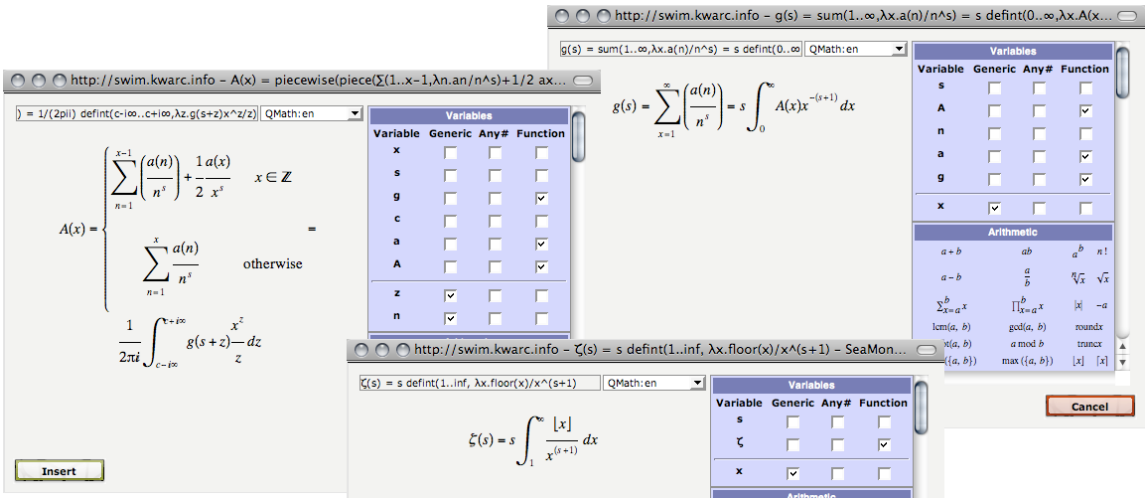


Figure 2: The formula editor window, when editing three different formulæ. The Variables palette allows to declare variables as functions. All symbols have Unicode and ASCII variants (∞/inf), and outermost parentheses do not need to be complete as seen in the bottom example.

Currently we provide four different syntaxes, but more can be defined via XML "context" files [8]. What those CAS syntaxes have in common, in contrast with LaTeX, is that they describe the content of mathematical expressions. Note that it is not possible to automate the translation from LaTeX syntax to content formulæ, as LaTeX is rather presentation-oriented, but an approximation would be possible.

It is not necessary to open the formula editor to do minor edits to the formulæ: any changes to the linear formula text are reflected in both the content of the formula editor if called later, and in the submitted OpenMath XML content.

Since the editor keeps track of the syntax used for each formula (displayed as tooltips), it is possible to have formulæ temporarily in different syntaxes while editing. However, the next time this page is opened for editing in SWiM, all formulæ will be translated from OpenMath to the same syntax as specified by the server. This way it is possible to quickly paste a formula in any of the supported syntaxes without having to convert the rest.

The formulæ are submitted as a string serialization of OpenMath XML, so that they do not interfere with TinyMCE or the browser. Using XML directly would corrupt the content because the editor works in HTML mode. On the server, this string is parsed back into XML. When a page is opened for editing next, the server again has to provide any contained OpenMath formulæ in their string serialization. Both are done in the same processing step as the conversion of the other CD markup to HTML tables. In this application of the formula editor, we can not display the MathML formulæ as is done in Sentido and other formula editors like AScience Pad [10] because of interference from other components.

Undo/redo inside the linear input field in the formula editor is provided by the browser, which is enough as changes in the text field are parsed back immediately, while outside it is handled by TinyMCE. Inside the formula editor each change can be undone/redone, but once we leave it the whole formula becomes an undo step.

4.2 Reacting to Changed Notation Definitions

Whenever the notation of a symbol σ has changed, all the presentation markup generated from formulæ containing σ has to be invalidated and re-rendered upon the next request. This

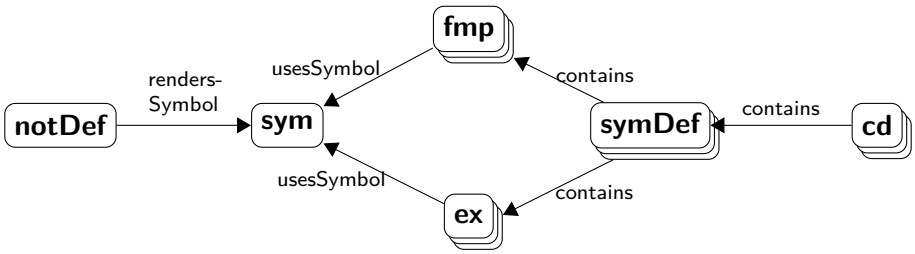


Figure 3: Finding pages (depicted as stacks of nodes) affected by changes to a notation definition. Both *sym* and the *symDef*s are instances of the class *SymbolDefinition*.

addresses the use case outlined initially. A notation definition for a symbol maps a pattern of content markup (a *prototype*) to a fragment of presentation markup (a *rendering*). For example, a notation definition for the root operator could look like $@(arith1\#root, \underline{\texttt{arg}}, \underline{\texttt{n}}) \vdash \sqrt[\underline{\texttt{n}}]{\underline{\texttt{arg}}}$[6]. From this, the RDF triple `<url/of/NotDef> omo:rendersSymbol <url/of/arith1/root>` would be extracted. Whenever a wiki page containing notation definitions is saved or imported, the notation definitions are put into a cache read by the *mmlproc* renderer.

If a notation definition has been added, deleted, or changed, the affected documents have to be re-rendered. In order to do this properly, SWiM has to (1) identify changes to notation definitions, and (2) identify documents affected by a change. (1) is done by computing an XML diff between the cached and the newly inserted version of a notation definition. (2) is done by querying the RDF graph for all FMPs and examples using the symbol rendered by the respective notation definition, as shown in fig. 3 and technically explained in [16][7]. Not only the wiki pages holding these FMPs and examples have to be re-rendered, but also those pages (symbol definitions and CDs) that directly or indirectly *include* these fragments.

5 Related work

The OpenMath edition of SWiM deals with semi-formal mathematical content. **ProofWiki** is a prototype of a wiki that contains fully formalized mathematical content [4]. The semantic structures of the content are, however, only used by the integrated Coq proof assistant, not to facilitate browsing or editing. Human-readable descriptive texts are written in non-semantic LaTeX. We are instead planning to build more support for the OMDoc language into SWiM, in order to support the full range between completely informal and completely formal mathematical knowledge.

OMPE (OpenMath Presentation Editor [19]) is an editor for **notation definitions** for OpenMath symbols. The content pattern to be matched is entered in OQMath, a variant of our linear QMath syntax. The resulting presentational pattern can be edited in a LaTeX-like syntax; common presentation symbols can be inserted via a toolbar. This makes it much more usable than SWiM in its current state. We are planning to address this by reusing parts of our formula editor, such as the parser for the linear input syntaxes, for editing notation definitions, too. On the other hand, debugging notation definitions by previewing their effect to rendered formulæ is only possible by *afterwards* feeding them to a presentation pipeline and viewing the rendered documents in the ActiveMath environment, whereas SWiM offers both in an integrated environment.

Managing changes to notation definitions has been investigated for the TEX_{MACS} editor

[6] In this abstract syntax, @ means an application of a symbol to arguments, and `underlined` variables are placeholders for subtrees that are rendered recursively [14]. Actually, all this is encoded in XML.

[7] Currently, (1) is implemented. Queries as needed for (2) work, but the IkeWiki system underlying SWiM [16] still flushes the whole cache of rendered pages when a page is saved.

before, which has been extended towards semantic markup in the PLATΩ project [2]. The developers focus on notations that use natural language and on parsing text and formulæ the user writes in a presentational style back to a semantic representation. Both features have not yet been investigated in SWiM; here the focus is rather on making the semantic markup editable in a convenient way. As a change to a notation definition in PLATΩ/TEXMACS involves regenerating parser rules, special attention is paid to making this efficient by only regenerating those rules that are affected by a change.

Both WIRIS [7] and ASciencePad [10] feature a **visual formula editor** integrated into HTMLArea, an editing widget similar to TinyMCE. ASciencePad is an extension of the single-file and single-user "wiki" TiddlyWiki; its math editor translates a linear syntax to Presentation MathML. WIRIS have instead integrated a Java applet for visually editing Presentation MathML formulæ and offer this as a plugin for the Moodle e-learning platform. No linear input syntax is used, but the formula is composed, not only previewed, two-dimensionally, and inside HTMLArea the formulæ are previewed as images. However, neither of the two editors edits content markup. WIRIS have developed an OpenMath editor [6], but that one is not integrated into HTMLArea.

6 Conclusion and Outlook

We presented how SWiM facilitates the editing of OpenMath CDs, particularly by integrating a visual formula editor, and by immediately re-rendering documents affected by notational changes thanks to the underlying graph of semantic relationships between parts of the CDs. Thus, it does not only facilitate CD editing, but editing any mathematical documents that uses symbols from OpenMath CDs. Now that SWiM has a reasonable coverage of the OpenMath 2 standard, plus the notation definition syntax upcoming with OpenMath 3, we hope to deploy it as a platform for browsing and maintaining the OpenMath 2 CDs at `http://www.openmath.org` soon. Once the details of the OpenMath 3 CD format are settled, we will adapt SWiM to these, updating both the ontology and the user interface, and deploy SWiM as an editor for the OpenMath 3 CDs. We anticipate more feedback about usability from that first public deployment after the release of the live demo at `http://swim.kwarc.info`.

So far, SWiM assumes one notation definition per symbol. *mmlproc* supports callbacks to an algorithm that selects the most appropriate out of a set of multiple possible renderings for a symbol [14]. In future, it is planned to provide a user interface inside SWiM that lets the user select his preferred rendering for every symbol. While SWiM supports *browsing* CDs well, with typed navigational links and symbols in formulæ linked to their definitions, *searching* formulæ is not yet supported. Search could be provided by the MathWebSearch engine [13], which would be instructed to crawl SWiM's database of documents. Editing CDs and their subparts currently works best if an existing CD is imported and then split automatically. We are aware of a demand for better refactoring support *within* SWiM [17], which remains to be implemented.

We are still keeping in mind that SWiM should not only be a collaborative editor for OpenMath, but also for the more expressive but more complex OMDoc language [12]. The achievements made for OpenMath support in SWiM in terms of knowledge representation, browsing, editing, and import/export are currently being transferred to OMDoc in order to improve the OMDoc support over the state presented earlier [15]. Thus, user feedback obtained from deploying SWiM to the OpenMath community will also lead to improvements for OMDoc users. Finally, with the Sentido Firefox plugin [9] we have developed an advanced WYSIWYG editor for OMDoc. We are planning to integrate it more tightly with SWiM and thus offer it as a powerful alternative editor to our extended TinyMCE.

References

[1] M. A. Abánades, J. Escribano, and F. Botana. First steps on using openmath to add proving capabilities to standard dynamic geometry systems. In Kauers et al. [11], pages 131–145.

[2] S. Autexier, A. Fiedler, T. Neumann, and M. Wagner. Supporting user-defined notations when integrating scientific text-editors with proof assistance systems. In Kauers et al. [11].

[3] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, and M. Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. `http://www.openmath.org/standard/om20`.

[4] P. Corbineau and C. Kaliszyk. Cooperative repositories for formal proofs. In Kauers et al. [11].

[5] J. H. Davenport and P. Libbrecht. The Freedom to Extend OpenMath and its Utility. *Journal of Mathematics and Computer Science, special issue on Mathematical Knowledge Management*, 2008.

[6] R. Eixarch. WIRIS editor. `http://www.wiris.com/content/view/20/`.

[7] R. Eixarch. WIRIS plugin for Moodle. `http://www.wiris.com/content/view/96/`.

[8] A. González Palomo. QMath: A human-oriented language and batch formatter for OMDoc. In OMDoc – *An open markup format for mathematical documents [Version 1.2]* [12].

[9] A. González Palomo. Sentido: an authoring environment for OMDoc. In OMDoc – *An open markup format for mathematical documents [Version 1.2]* [12].

[10] P. Jipsen. ASciencePad – a TiddlyWiki suitable for scientific notes. `http://math.chapman.edu/~jipsen/asciencepad/asciencepad.html`.

[11] M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors. *MKM/Calculemus 2007*, number 4573 in LNAI. Springer, 2007.

[12] M. Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer, 2006.

[13] M. Kohlhase, Ş. Anca, C. Jucovschi, A. González Palomo, and I. A. Şucan. MathWebSearch 0.4, a semantic search engine for mathematics. Manuscript at `http://mathweb.org/projects/mws/pubs/mkm08.pdf`, 2008.

[14] M. Kohlhase, C. Müller, and F. Rabe. Notations for living mathematical documents. In *Mathematical Knowledge Management, MKM'08*, LNAI. Springer, 2008. in press.

[15] C. Lange. SWiM – a semantic wiki for mathematical knowledge management. In P. Libbrecht, editor, *Mathematical User Interfaces Workshop*, 2007.

[16] C. Lange. Mathematical Semantic Markup in a Wiki: The Roles of Symbols and Notations. In Lange et al. [18].

[17] C. Lange, T. Hastrup, and S. Corlosquet. Improving mathematical knowledge items by acting on issue-based community feedback. In C. Müller, editor, *2nd SCooP Workshop*, 2008.

[18] C. Lange, S. Schaffert, H. Skaf-Molli, and M. Völkel, editors. *3rd Workshop on Semantic Wikis*, 2008.

[19] S. Manzoor, P. Libbrecht, C. Ullrich, and E. Melis. Authoring Presentation for OpenMath. In M. Kohlhase, editor, *Mathematical Knowledge Management, MKM'05*, number 3863 in LNAI. Springer, 2005.

[20] M. Marchiori. The mathematical semantic web. In A. Asperti, B. Buchberger, and J. H. Davenport, editors, *Mathematical Knowledge Management, MKM'03*, number 2594 in LNCS. Springer, 2003.

[21] J. Marsh, D. Orchard, and D. Veillard. XML inclusions (XInclude) version 1.0 (second edition). Recommendation, W3C, Nov. 2006. `http://www.w3.org/TR/2006/REC-xinclude-20061115/`.

[22] OpenMath content dictionaries. `http://www.openmath.org/cd/`.

[23] TinyMCE – a platform independent web based JavaScript HTML WYSIWYG editor. `http://tinymce.moxiecode.com/`.