

Proceedings of the
ITP 2011 Workshop on
Mathematical Wikis (MathWikis 2011)
Nijmegen, Netherlands, August 27, 2011

edited by Christoph Lange and Josef Urban

August 27, 2011

Preface

This volume contains the papers presented at MathWikis (<http://www.cs.ru.nl/mwitp/>), the ITP 2011 Workshop on Mathematical Wikis held on August 28, 2011 in Nijmegen, Netherlands.

There were 10 submissions (one of them invited). Each submission was reviewed by four program committee members.

We would like to thank Henk Barendregt for generously sponsoring the MathWikis workshop from his Spinoza Prize, awarded by The Netherlands Organisation for Scientific Research, as well as our peer reviewers for carefully reviewing the submissions and giving constructive feedback.

This proceedings volume has been generated with EasyChair, which made this task really convenient.

August 23, 2011
Bremen/Nijmegen

Christoph Lange
Josef Urban

Program Committee

Jesse Alama	CENTRIA, FCT, Universidade Nova de Lisboa (PT)
David Aspinall	University of Edinburgh (UK)
Joseph Corneli	Knowledge Media Institute, The Open University (UK)
Cezary Kaliszyk	University of Tsukuba (JP)
Fairouz Kamareddine	Heriot-Watt University (UK)
Michael Kohlhase	Jacobs University Bremen (DE)
Markus Krötzsch	University of Oxford (UK)
Christoph Lange	Jacobs University Bremen (DE)
Lionel Mamane	(LU)
James Mckinna	Radboud University Nijmegen (NL)
Piotr Rudnicki	Univ. of Alberta (CA)
Carst Tankink	Radboud University Nijmegen (NL)
Josef Urban	Radboud University Nijmegen (NL)
Denny Vrandečić	Karlsruhe Institute of Technology (DE)

Contents

Preface	ii
Programme	v
The On-Line Encyclopedia of Integer Sequences: From Punched Cards to Wiki in 46 Years Neil J. A. Sloane	1
Metadata for a wiki of formalized mathematics Jesse Alama	2
The PlanetMath Encyclopedia Joseph Corneli	6
A Linear Algebra Wiki Michael Doob	13
The Web of Mathematical Models Thomas Grundmann, Jean-Marie Gaillourdet, Karsten Schmidt, Arnd Poetzsch-Heffter, Stefan Deßloch, and Martin Memmel	19
Wiki Authoring and Semantics of Mathematical Document Structure Hiraku Kuroda and Takao Namiki	28
Ideas for a MathWiki Editor Sebastian Reichelt	38
Dynamic Proof Pages Carst Tankink and James McKinna	45
Content-based encoding of mathematical and code libraries Josef Urban	49
ProofWiki: A Structured Approach to Mathematical Presentation Matt Westwood	54
WorkingWiki: a MediaWiki-based platform for collaborative research Lee Worden	63

Programme

Contributed talks are 30 minutes long. Please allow at least 5 minutes of your time slot for discussion.

09:00–10:00 Session 1

9:00–10:00 Invited talk: The PlanetMath Encyclopedia
Joseph Corneli

10:00–10:30 **Coffee Break**

10:30–12:30 Session 2

10:30–11:00 The Web of Mathematical Models: A Schema-based, Wiki-like, Interactive Platform
Thomas Grundmann, Jean-Marie Gaillourdet, Karsten Schmidt, Arnd Poetzsch-Heffter, Stefan Deßloch and Martin Memmel

11:00–11:30 ProofWiki: A Modular Approach to Mathematical Presentation
Matt Westwood

11:30–12:00 Wiki Authoring and Semantics of Mathematical Document Structure
Hiraku Kuroda and Takao Namiki

12:00–12:30 Metadata for a mathematical wiki: Initial experiments
Jesse Alama

12:30–14:00 **Lunch Break**

14:00–15:30 Session 3

14:00–15:00 Invited talk: The On-Line Encyclopedia of Integer Sequences: From Punched Cards to Wiki in 46 Years.
Neil J. A. Sloane

15:00–15:30 A Linear Algebra Wiki
Michael Doob

15:30–16:00 **Coffee Break**

16:00–18:00 Session 4

16:00–16:30 WorkingWiki: a MediaWiki-based platform for collaborative research
Lee Worden

16:30–17:00 Dynamic Proof Pages
Carst Tankink and James McKinna

17:00–17:30 Ideas for a MathWiki Editor
Sebastian Reichelt

17:30–18:00 Content-based encoding of mathematical and code libraries
Josef Urban

The On-Line Encyclopedia of Integer Sequences: From Punched Cards to Wiki in 46 Years

Neil J. A. Sloane
AT&T Shannon Labs,
Florham park, NJ, USA

Abstract

I started collecting number sequences in 1964. In 1996 I launched the On-Line Encyclopedia of Integer Sequences (the OEIS) on the web, with 10000 sequences. By 2008 it had grown to 135000 sequences, and I was processing 200 emails a day - too much for one person to handle. But converting it to a Wiki was very difficult and took two years. Only thanks to the efforts of Russ Cox, who had to rewrite all the OEIS software, and with the help of my colleague David Applegate, the OEIS Wiki finally started working on November 11 2010 (see <http://oeis.org>). In this talk I will report on the problems we ran into, how we solved them, and the present state of the OEIS Wiki.

Metadata for a wiki of formalized mathematics

Jesse Alama*

Center for Artificial Intelligence

New University of Lisbon

j.alama@fct.unl.pt

Abstract

In recent years wikis for formal mathematics have appeared. Formal mathematics presents a number of challenges for the wiki perspective. To enhance the quality of the data in these wikis from the perspective of information architecture, we propose some extensions of existing formal mathematics wikis to more properly handle *metadata*.

1 Introduction

Recent years have seen some attention paid to the problem of building wikis for formalized mathematical texts, and there are various proposals and even some live wiki or wiki-like systems for formal mathematics [9, 6]. Formal mathematics (or indeed any formally verified content) presents a number of challenges for the wiki perspective.

In this paper we propose some extensions of existing formal mathematics wikis to more properly handle *metadata*. The aim, ultimately, is to better apply the tools of *information architecture* [7] for a formal mathematical wiki. We sketch some salient kinds of metadata for formal mathematics and their implementation. Our work is but a modest step toward the grand aim of a semantic web in the domain of mathematics, with a bias toward formal verification.

2 Previous work

Our implementation concerns mizar, primarily, though our proposals naturally extend to other interactive theorem provers and indeed other formal wikis as well.

mizar is an interactive theorem prover built on classical first-order logic and set theory. Its associated library, the mizar Mathematical Library (MML), is a large corpus of formalized mathematical knowledge spanning many areas of mathematics.

One early resource is the mizar wiki, at <http://wiki.mizar.org>. This is a long-standing resource for those interested in the mizar language and its library. Part of that wiki documents metadata for the articles of the MML. Of interest to us are the tags assigned to articles from the American Mathematical Association's Mathematics Subject (MSC) Classification scheme.

However, the mizar wiki does not support editing of other metadata about articles of the library, and it serves its (HTML) resources without contentful metadata, neither in the HTTP headers nor in the representations themselves. The first wiki system for mizar is live at <http://mws.cs.ru.nl/mwiki>. This wiki does support editing mizar texts and ensures their validity. But this wiki also offers up only scant metadata. We build on earlier wiki efforts for formalized mathematics [9, 2], and on a dynamic website [1] for presenting fine-grained logical and mathematical dependencies for the mizar Mathematical Library. Our aim is to extend the infrastructure provided by these systems with rich metadata, thereby increasing the quality of mizar information available to humans and machines alike.

*The author was supported by the FCT project "Dialogical Foundations of Semantics" (DiFoS) in the ESF EuroCoRes programme LogICCC (FCT LogICCC/0001/2007).

The efforts of Kohlhase and his collaborators (e.g., [6]) toward semantic markup of mathematics, though not about formalized mathematics in the same sense as mizar texts, have helped to show ways of marking up mathematical texts.

3 Metadata categories

Some metadata about mizar texts are functions of the text—we can compute the metadata in question—and thus aren't suitable for user editing. The length of a mizar article in bytes, for example, or the underlying XML representation of an article, are properties of the text that cannot (so far as we can see) be meaningfully modified.

Other properties of texts, however, can be profitably exposed. The following properties of mizar texts are candidates for user-editable properties:

- Mathematics Subject Classification

Already there has been some informal efforts to annotate parts of the mizar Mathematical Library by assigning tags to articles from the American Mathematical Society's widely used Mathematical Subject Classification (MSC) scheme [4]. These data are used to seed our MSC assignments: we reuse the previous assignments of MSCs to mizar items (generally whole articles) for our initial assignment.

- Title and author information.
- Natural language abstract.
- Citations of relevant mathematical sources (e.g., books, papers, encyclopedia entries).
- Uncomputed or undetectable relationships between other items.

The mizar definition of cardinal numbers, for example, depends implicitly on the axiom of choice, but it is not clear how this can be detected from the dependency graph of the mizar Mathematical Library (if it is possible at all). It seems better to us to permit users of the mizar wiki to associate the mizar item that represents the definition of cardinal number with the mizar item that represents the axiom of choice. Such links could be represented through Atom link elements and served as See-Also HTTP headers.

- Alternates for theorems and definitions

Thus, a “named” theorem might have different variants or special cases. For example, the Jordan curve theorem can be understood in complete generality, or in the special case of polygons. A theorem might be stated for arbitrary dimensions, but be of especial interest for dimension 2 or 3 (e.g., Euler's polyhedron formula in full generality and the three-dimensional case). A theorem (or axiom, as the case may be) can even have different forms, such as Zermelo's well-ordering principle (“every set can be well-ordered”) and a ‘vanilla’ rendering of the axiom of choice (“every non-empty set of non-empty sets has a choice function”) are equivalent to one another (in a suitable background theory, of course).

Some of these metadata categories are already handled outside the context of our wiki. When authors submit their mizar texts to SUM (the mizar User Group, charged with maintaining mizar and its library), they supply the text as well as author and title information, an abstract (in English), and citations to relevant sources (e.g., the source of the “informal” proofs that are formalized in the article). This information is used to automatically generate a T_EX representation of an article, which is then published in the

journal *Formalized Mathematics*¹. The editors of *Formalized Mathematics* are charged with maintaining this information. The data is thus static, in the sense that one cannot edit the published title and abstract of a paper that has already been published in a journal.

A wiki that permits editing of metadata that already exists under the aegis of *Formalized Mathematics* is not intended to compete with that journal. The SUM and its efforts to professionalize the writing of formal mathematical texts (in mizar) complement the project proposed here. We use the mass of metadata that has been accumulated throughout the construction of the MML to seed our wiki with an initial pool of trustworthy data. Our aim is to make the sea of (mizar-)formalized mathematical knowledge discoverable and improvable. Where possible, we provide links to the authoritative, original sources for mizar formalizations.

Metadata support in the mizar language itself is (almost) entirely lacking. One cannot use the language itself to indicate, for example, the author of a mizar text. The language supports only one metadata-related construction, the `section` keyword, which takes no arguments and it intended to delimit different parts of an article.

In the context of a live wiki, however, curation-by-committee is not a viable option for eliciting and maintaining metadata. One must go beyond the mizar language and provide a means of attaching or editing metadata. Since one cannot directly add this information to mizar texts, in the context of a web interface, one must enter the information in a form.

The absence of metadata functionality from the mizar language (apart from the aforementioned `section` keyword) is, to some extent, an advantage, because it gives a clear separation of responsibility of the validity of the text from information about the text. Our wiki process for registering and maintaining metadata can complement the standard process for eliciting this data. With each new release of the mizar Mathematical Library, we receive another batch of “canonical” or official metadata for updated contents of the library.

4 Serving metadata

We serve metadata in two ways: in the HTTP headers, and in the body of our article representations. We adhere to RESTful principles [3], promoting the transparency of the HTTP protocol with the ultimate aim of making our (meta)data accessible to humans and machines (e.g., search engines) alike.

We use the Atom Publishing Protocol and its extensible notion of link [8] by serving metadata in the HTTP response headers. Our aim coheres with the goals of the Linked Open Data initiative². Given a request for, say, the 92nd theorem of the article POLYFORM, we can return the following headers:

```
# Request
GET /item/polyform/theorem/92 HTTP/1.1

# Response
HTTP/1.1 200 OK
Content-Type: text/html
Link: <http://wiki.example.org/item/polyform/theorem/92/text>;
    rel="alternate";
    type="text/plain"
Link: <http://wiki.example.org/authors/5>;
    rel="http://wiki.example.org/rels/author"

...
```

¹See <http://fm.mizar.org>.

²See <http://linkeddata.org/>

In this example, in addition to standard HTTP headers, we use the proposed Link HTTP header [5] to refer to the author of the article (if it is known). The representation in the . . . would likewise contain a link in its body to the author and the other metadata.

One challenge in the context of a wiki is to provide keep URIs “cool”, that is, essentially permanent objects. The slogan here is “Cool URIs don’t change” [3]. Metadata can help us to keep URIs cool while still admitting that some of the concepts, definitions, and theorems to which the URIs refer are evolving.

5 Outlook

Richer representations of formalized mathematical data and its metadata are available. Thus, one could serve RDFa or employ a suitable microdata framework. Richer representations of the mizar data itself as RDFa, or a documented, well-structured URI namespace for items, would be valuable. In the face of capable clients, one could even imagine performing non-trivial content negotiation; today, mizar text are served only as plain text or as (relatively thinly annotated) HTML.

Finally, it should be clear that although we focus on mizar, few parts of the project that directly depend on the mizar language, its tool set, or its library. The wiki principles developed here would apply to any other system as well. Indeed, a formal wiki for other systems, suitably equipped with the kind of metadata, would be a step toward setting up rich correspondences between various formal libraries and their web representations.

References

- [1] Jesse Alama. mizar-items: Computing fine-grained dependencies in the mizar mathematical library. To appear in *CICM 2011: Conference on Intelligent Computer Mathematics*.
- [2] Jesse Alama, Kasper Brink, Lionel Mamane, and Josef Urban. Large formal wikis: Issues and solutions. submitted, 2011.
- [3] Subbu Allamraju. *RESTful Web Services Cookbook*. O’Reilly, Sebastopol, California, 2010.
- [4] American Mathematical Society. *Mathematics Subject Classification*, 2010.
- [5] D. Connolly and I. Hickson. An entity header for linked resources. Technical report, Internet Engineering Task Force, April 1999. <http://www.w3.org/Protocols/9707-link-header.html>.
- [6] Christoph Lange and Michael Kohlhase. A semantic wiki for mathematical knowledge management. In Jörg Rech, Björn Decker, and Eric Ras, editors, *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications*, pages 47–68. IGI Global, April 2008.
- [7] Peter Morville and Louis Rosenfeld. *Information Architecture for the World Wide Web*. O’Reilly, Sebastopol, California, 3 edition, 2007.
- [8] The Atom publishing protocol. Available online at <http://tools.ietf.org/html/rfc5023>, October 2007.
- [9] Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. A wiki for mizar: Motivation, considerations, and initial prototype. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *Intelligent Computer Mathematics, 10th International Conference, AISC 2010, 17th Symposium, Calculemus 2010, and 9th International Conference, MKM 2010, Paris, France, July 5-10, 2010. Proceedings*, volume 6167 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2010.

The PlanetMath Encyclopedia

Joseph Corneli
Knowledge Media Institute
The Open University
Milton Keynes, UK
j.a.corneli@open.ac.uk

Abstract

The history of PlanetMath.org is discussed, tracing its inception, stabilization, and some defining challenges. Research and outreach efforts that have been conducted in the course of work on the PlanetMath project are reviewed, and the scope and reach of the resource are discussed. Recent developments are indicated briefly. Some remarks evaluating PlanetMath's trajectory and content conclude the paper.

Keywords: online communities, mathematics, collaboration, encyclopedias, Commons-Based Peer Production, PlanetMath

1 Introduction

From PlanetMath.org's landing page¹:

PlanetMath is a virtual community which aims to help make mathematical knowledge more accessible. PlanetMath's content is created collaboratively: the main feature is the mathematics encyclopedia with entries written and reviewed by members.

This short paper describes the history of the PlanetMath encyclopedia. The history of this resource cannot be easily separated from a history of the PlanetMath community and the technology behind the site, though the presentation here is not especially technical. The reader who is interested in a succinct overview of the *current* characteristics of the encyclopedia will find what they seek in Sections 6 (quantitative aspects) and 8 (qualitative aspects).

2 Beginnings

The early history of PlanetMath is wrapped up with that of the similarly-named website, MathWorld.²

Eric Weisstein began collecting the material now found in MathWorld as a high school student, and continued the project as a college student in the late 1980s. "Eric's Treasure Trove of Mathematics," went online in 1995, when Weisstein was a graduate student in astronomy at the California Institute of Technology.³ In November 1998, Weisstein made a deal with the CRC Press to publish his encyclopedia in book format, as the *CRC Concise Encyclopedia of Mathematics*. One year later, Weisstein accepted the position of encyclopedist at Wolfram Research, Inc., and the renamed "MathWorld" site was unveiled in December 1999.⁴ In March 2000, CRC Press sued Weisstein and Wolfram Research for copyright violation, forcing MathWorld off of the internet.⁵

¹<http://planetmath.org>

²<http://mathworld.wolfram.com>

³<http://www.echarcha.com/forum/archive/index.php/t-19516.html> a copy of the now-defunct http://mathworld.wolfram.com/about/erics_commentary.html

⁴http://en.wikipedia.org/wiki/Eric_W._Weisstein

⁵57 U.S.P.Q.2d 1220 (C.D. Ill. 2000)

In the words of Eric Weisstein: “if you ever assemble a body of knowledge that you want to share with others, you don’t want to go through what I have just gone through.”⁶ So it came to pass that in the Fall of 2000, Nathan Egge and Aaron Krowne, at that time both undergraduates at Virginia Tech, came up with the idea for PlanetMath: a collaboratively created mathematics reference work that would have resistance to copyright threats built in, in the form of an open content license. By the summer of 2001, the basic infrastructure for creating an encyclopedia was complete, and a fledgling community had grown up around the resource.

The CRC lawsuit was settled for an undisclosed sum in late 2001, and on November 6, 2001, MathWorld returned to the internet.⁷ But in the mean time, a new online community had been born – with some very different principles and practices. Whereas MathWorld’s terms of use disallow archival copies, PlanetMath regularly publishes snapshots of the content for download. Moreover, users are permitted (and, indeed, encouraged) to copy, mirror, redistribute, print, remix, and reuse PlanetMath content for commercial or any other purpose – so long as all such works are published under the same license as PlanetMath, granting downstream users the same rights.

3 Stabilization

The key reference for PlanetMath is Aaron Krowne’s 2003 Master’s Thesis, written at Virginia Tech [6] under the supervision of Ed Fox. In this thesis, Krowne describes how the early design and development of the site benefited from continuous feedback in the #math IRC channel on Undernet.⁸ He also details the key technical and community features of the site as they developed in this period:

- A state-of-the-art system for displaying mathematical notation on the web, starting from L^AT_EX sources.
- A flexible authority model that can support both wiki-style articles (that anyone can edit), and a more academic style, where articles are owned by one person, who may, if they wish, grant co-authorship permissions to chosen others, and who must respond to separate commentary from peer reviewers (cf. [7]).
- A discussion forum attached to every encyclopedia article, which helps give the resource a “pedagogical slant”.
- An autolinking service that helps integrate content into the site, by enabling authors to focus on the contents of one article at a time.
- Workflow built around corrections and watches, including a feature whereby articles are “orphaned” if a correction is not responded to after a given period of time.
- A scoring feature that provides a rough estimate of how much value each user has contributed to the site.

In 2003, PlanetMath incorporated, and in 2005, obtained non-profit status, so that it could accept tax-deductible donations (in the US). Together with a small stream of ad revenue, this has covered hosting and other maintenance costs.

⁶See Footnote 3.

⁷<http://mathworld.wolfram.com/about/faq.html#history>

⁸<irc://irc.undernet.org/math>

4 Pushing the limits

In 2003, the present author was enrolled as a graduate student in mathematics at the University of Texas in Austin, and in possession of a large and growing personal collection of very tersely-written definitions and proofs relevant to the department’s prelim exams.⁹ In fact, this work had as much to do with the tradition of computer mathematics in the air in Austin (QED, Maxima, ACL2, AM) as it had to do with exams. A representative example:

(lebesgue outer measure: fact: lebesgue outer measure is infimum of lebesgue outer measures of open supersets)

- 1: $X \subset \mathbf{R}^n$
- 2: $L = \{O \subseteq \mathbf{R}^n : O \supset X\}$
- 3: $|X|_e = \inf_{O \in L} (|O|_e)$

After discovering PlanetMath and striking up a correspondence with Krowne, one night we uploaded the contents of the “Austin Problems in Mathematics – Cross-Index” (styled APM- ξ) into the PlanetMath encyclopedia as world-editable “seed entries”. This turned out to be a first-rate disaster.¹⁰

The primary complaints from community members were:

- (1) the entries could not be understood without reading an accompanying FAQ;
- (2) a casual visitor to the PlanetMath website might get the wrong impression about the nature of the encyclopedia when looking at “apmxi” entries; and,
- (3) nearly 600 entries had been introduced into PlanetMath by the site’s administrator in one big batch, circumventing, at least in outward appearances, the site’s usual model of careful review and collaborative editing of entries.

Subsequent to a poll, it was decided that the apmxi entries would be “orphaned”, and any that were not adopted by community members after a week would be deleted from the encyclopedia. This was the fate that befell most.

The event was a defining moment in the history of PlanetMath. In the first place, it was a testament to the strength of the community’s norms. Secondly, it showed that the specific affordances of computers, e.g. for mass processing of data, or for dealing with hypertextual complexity associated with alternate related treatments of a given topic, needed to be tempered to work well for the *people* involved. These issues would set much of the research and development agenda around PlanetMath for the following decade.

5 Research, outreach, and some critiques

In 2005, several established PlanetMath contributors met in person at a Symposium on *Free Culture and the Digital Library* at Emory University, where Krowne was then based. Contributed papers looked at

- an adaptation of PlanetMath’s software for collaborative creation of course notes in a graduate course on ordinary differential equations [10];
- experiments with a novel hypertext system based on the idea that everything is annotatable [1]; and,

⁹<http://metameso.org/~joe/math/Xi.pdf>

¹⁰http://wiki.planetmath.org/cgi-bin/wiki.pl/one_week_in_october

- the dynamic tension between the non-copyrightability of ideas, and the necessity of conveying ideas in copyrightable expressions, and the ramifications for mathematics [12].

These reflections on copyright (and copyleft) were subsequently expanded in an article for *First Monday*, which looked at the drawbacks of current copyleft licenses, particularly “license lock” [8].

We presented talks about PlanetMath in the *Math on the web pavilion* at two Joint Mathematics Meetings (San Antonio, 2006¹¹; New Orleans, 2007¹²), and in a session on *The Role of Open Source Math Projects in the Mathematics Community* at MathFest (Madison, 2008)¹³; and at more specialized workshops: *The Evolution of Mathematical Communication in the Age of Digital Libraries* (Minneapolis, 2006)¹⁴, and *Mathematical Knowledge Management: Sustainability, Scalability and Interoperability* (Halifax, 2007)¹⁵.

We also made efforts to create a print version of the PlanetMath encyclopedia (retitled the “Free Encyclopedia of Mathematics”). The 2004 attempt, in two volumes¹⁶, and a 2005 attempt in one much nicer-looking volume¹⁷, thanks to Ross Moore’s contribution of `multinclude.sty` and other tweaks.¹⁸ Still, the resulting 1971 page PDF was more a proof of concept than a printer’s proof.

On the development side, PlanetMath was thrice supported by Google’s Summer of Code (2006–2008). The best outcome of this was that PlanetMath’s autolinking subsystem was turned into a modular piece of code, NNexus¹⁹, as written up in [5]. PlanetMath’s software improved further under contract with Springer, pursuant to the creation of StatProb.com.²⁰ PlanetMath’s sister site PlanetPhysics.org²¹ is currently in the process of switching over to this platform, termed Noosphere 1.5.

However, the development effort wasn’t particularly able to keep pace with the feature requests generated by the user community.²² Nor did the Noosphere codebase present a particularly compelling resource for capable computer mathematics developers like Claus Zinn [14] and Christoph Lange [9], to jump into and improve. Zinn wrote:

The rapid growth of math resources on the web, which is further pushed by wiki-based communities, is both a threat and an opportunity for intelligent math learning environments [...] If we could harness the collaborative authoring process and encourage and guide wiki authors to continually provide content and metadata, then intelligent services could unleash their true potential.

6 Scope and reach

At the time of this writing, PlanetMath contains 8945 entries, dealing with 15655 concepts. 298 people have contributed an entry in the encyclopedia, and 2742 have contributed something (perhaps just one forum post).

¹¹www.jointmathematicsm meetings.org/meetings/national/jmm/san-prog.pdf

¹²<http://www.dessci.com/en/company/shows/jmm/mow2007.htm>

¹³<http://www.maa.org/abstracts/mf2008-program.pdf>

¹⁴<http://www.ima.umn.edu/2006-2007/SW12.8-9.06/>

¹⁵<http://projects.cs.dal.ca/ddrive/seminars/mkm.shtml>

¹⁶<http://www.scribd.com/doc/9691966/>, <http://www.scribd.com/doc/9692058/>

¹⁷<http://metameso.org/~joe/docs/book.pdf>

¹⁸<http://metameso.org/~joe/math/fem-2005.tar.gz>

¹⁹<http://code.google.com/p/nnexus/>

²⁰<http://statprob.com>

²¹<http://planetphysics.org>

²²http://wiki.planetmath.org/cgi-bin/wiki.pl/Feature_Requests

Out of these, an exceptional group of 24 authors have produced more than 100 encyclopedia articles apiece. Their contributions comprise 74% of the total number of articles. About 71% of this core group joined before 2004 (in the “early days” for the site).

130 users have a score of 1000 points or more, which would correspond to contributing 10 or more new encyclopedia articles, but actually, a significant fraction of this value has been contributed through things like corrections, revisions to existing objects, and posting in the forums. All told, this group has contributed 96% of the total number of articles. About 58% of this group joined before 2004.

According to Alexa.com, PlanetMath.org is currently the 165,011th most popular website in the world, comparable to the website of the Mathematical Association of America²³ (119,267th), or relative newcomer MathOverflow.net²⁴ (184,818th).

A far cry from competing with Wikipedia, but in fact we like to think of the two projects as mutually supporting. PlanetMath content is reused under the terms of the shared CC-By-SA license in hundreds of Wikipedia articles, and cited in over 1500.

7 A new era

In 2010, a new project to completely rebuild PlanetMath’s software began. *Planetary* is based at Jacobs University, Bremen, and led by Michael Kohlhase, with major contributions from most members of his research group (along with the present author).²⁵ The Planetary system is described in [4]. Planetary is considerably easier to extend than Noosphere: it is currently comprised of around 20 plugins for the popular open source platform, Vanilla Forums²⁶, many of which integrate sophisticated software tools previously developed by the KWARC group. Notably, the system now includes support for semantic authoring and flexible metadata interaction, addressing the critiques mentioned in Section 5.

2011 will see the publication of a book chapter discussing the future use of PlanetMath as the core of a problem-based learning system [2], the focus of the author’s doctoral studies [3]. With this as a basis for a showcase of innovative uses for the PlanetMath content, and with a well-documented and easy to extend software platform supporting the system, we hope to see PlanetMath become a central integration platform for free software projects working with freely licensed math on the web.

8 Conclusion

PlanetMath has been successful as an online community: the software stabilized early on and has required little upkeep, while the site has continued to grow. However, there has been a danger that with the software system running more or less on “autopilot”, new features would not be developed. With any luck, this threat is in the process of being eliminated, heralding in the opportunity to build one or more “new” online communities in close relationship to PlanetMath (e.g. a developer community working on sophisticated tools for scientific communication; a learning community using the PlanetMath encyclopedia as part of a remix-driven interactive textbook). We should do a careful evaluation of what has worked and what could be improved for next time. There is not room in the current paper to conduct this discussion, but framework proposed by Resnick *et al.* would be an excellent place to begin [13].

For all of its potential as a software showcase and its possible future role as a player in a larger landscape of related interlinked online communities, PlanetMath should, at least for the moment, be evaluated first and foremost as an encyclopedia. This too would be best as an ongoing task. For now,

²³<http://maa.org>

²⁴<http://mathoverflow.net>

²⁵<http://trac.mathweb.org/planetary>

²⁶<http://vanillaforums.org>

Coverage	The median entry would be an advanced undergraduate or beginning graduate topic. PlanetMath is generally considered to have more in-depth treatment of technical issues, e.g. of proofs, than Wikipedia.
References	Present in many articles, although there is not yet a unified database of references or style of presenting them (this is planned).
History	315 items, mostly 20th Century or later. ²⁷
Audience	Consistent with the coverage, there have been 4127 posts in the “Graduate/Advanced” forum, 4261 posts in the “University/Tertiary” forum, and 1199 posts in the “High School/Secondary” forum.
Clarity	There is no hard and fast rule. Some articles are minimalistic, but precise (e.g. from strong contributors who have English as a second language). Other articles may be verbose and vague. In any case, debates over clarity of presentation are intense, and a high standard is generally maintained (see Section 4).
Pictures	There are over 600 images, but this is only about 7% of all entries. A unified database/gallery of pictures might encourage more submissions.
Accuracy	At the time of this writing there are 48 outstanding corrections ²⁸ ; more than 14080 corrections have been filed since the site began, though 2337 are classified as “addenda”, meaning that no mistake is implied, and 9182 are classified as “meta/minor”, meaning that in the entire history of PlanetMath some 2561 real errors have been found through the peer review process (and all but a few fixed!). Note that these numbers do not take into account direct changes by authors (and would tend to under-represent error fixing in world-editable articles).
Unusual	PlanetMath provides “math for the people, by the people”.
Weight	PlanetMath can be used from any browser, and comfortably edited from a lightweight laptop or netbook, weighing about 1kg.

Table 1: Succinct review of the PlanetMath Encyclopedia

a quick summary following the outline used by Emma Previato in her review [11] of the *CRC Concise* gives us a look at how PlanetMath measures up (Table 1).

Because the technology that supports the site is special-purpose, we have been able to hone in on what works best for commons-based peer production in mathematics. Features like the autolinker facilitate integration of content, and the corrections system helps avoid messy battles. There is much more work to be done, but at the close of its first decade of life, PlanetMath may be poised to become a encyclopedia in the literal classical sense of a “complete instruction”.

²⁷<http://planetmath.org/browse/objects/01Axx/>

²⁸<http://planetmath.org/?op=globalcors>

References

- [1] J. Corneli and A. Krowne. A scholia-based document model for commons-based peer production. In M. Halbert, editor, *Free Culture and the Digital Library Symposium Proceedings*, pages 241–254, 2005.
- [2] J. Corneli and A. Mikroyannidis. Crowdsourcing education: A Role-Based analysis. In Alexandra Okada, Teresa Connolly, and Peter Scott, editors, *Collaborative Learning 2.0: Open Educational Resources*. IGI Global, 2011. (In press.).
- [3] Joseph Corneli. Problem solving and mathematical knowledge. Technical report, Knowledge Media Institute, 2010.
- [4] C. David, D. Ginev, M. Kohlhase, and J. Corneli. eMath 3.0: Building blocks for a social and semantic web for online mathematics & eLearning. In I. Mierlus-Mazilu, editor, *1st International Workshop on Mathematics and ICT: Education, Research and Applications*, Bucharest, Romania, 2010.
- [5] James Gardner, Aaron Krowne, and Li Xiong. NNexus: an automatic linker for collaborative web-based corpora. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 1152–1155, New York, NY, USA, 2009. ACM.
- [6] A. Krowne. An architecture for collaborative math and science digital libraries. Master's thesis, Virginia Polytechnic Institute and State University, 2003.
- [7] A. Krowne and A. Bazaz. Authority models for collaborative authoring. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS), 2004*. IEEE, 2004.
- [8] A. Krowne and R. Puzio. The fog of copyleft. *First Monday*, 11(7), 2006.
- [9] C. Lange. Towards scientific collaboration in a semantic wiki. In Andreas Hotho and Bettina Hoser, editors, *Bridging the Gap between Semantic Web and Web 2.0 (SemNet)*, 2007.
- [10] R. Milson and A. Krowne. Adapting CBPP platforms for instructional use. In M. Halbert, editor, *Free Culture and the Digital Library Symposium Proceedings*, pages 255–272, 2005.
- [11] E. Previato. Featured Review: CRC Concise Encyclopedia of Mathematics. *SIAM Review*, 46(2):349–374, 2004.
- [12] R. S. Puzio. On free math and copyright bottlenecks. In M. Halbert, editor, *Free Culture and the Digital Library Symposium Proceedings*, pages 273–299, 2005.
- [13] Paul Resnick, Joseph Konstan, Yan Chen, and Robert Kraut. Starting new online communities. In R. E. Kraut and P. Resnick, editors, *Evidence-based social design: Mining the social sciences to build online communities*. MIT Press, (Under contract).
- [14] C. Zinn. Bootstrapping a semantic wiki application for learning mathematics. In *International Conference on Semantics-Semantics 2006*, pages 27–29. Citeseer, 2006.

A Linear Algebra Wiki

Michael Doob
Department of Mathematics
The University of Manitoba
Winnipeg, Manitoba, Canada R3T 2N2
July 11, 2011

Abstract

This paper is a report on an ongoing project of two years. Its purpose is to provide a resource for Linear Algebra at the undergraduate level that can, in particular, be used as a textbook. The experiences, both good and bad are given, and these are used to indicate future directions for the project.

1 Motivation

1.1 Motivation #1: Correctability

The initial motivation for this project came from teaching an introductory course in Linear Algebra for first-year university students after having not done so for many years. I was really dissatisfied with the textbook. For example, the following “proof” was given:

Theorem 2.2.2 Let A be a square matrix. Then $\det(A) = \det(A^T)$.

Proof By Theorem 2.1.1, the determinant of A found by cofactor expansion along its first row is the same as the determinant of A^T found by cofactor expansion along its first column.

Putting aside the fact that Theorem 2.1.1 is described earlier in the text as a “general theorem, which we state without proof”, the purported proof itself is simply irreparably wrong in the absence of some kind of inductive argument.

Here is another sample from the same book:

Theorem 5.2.1 If W is a set of one or more vectors from a vector space V , then W is a subspace of V if and only if the following conditions hold.

- (a) If \mathbf{u} and \mathbf{v} are vectors in W , then $\mathbf{u} + \mathbf{v}$ is in W .
- (b) If k is any scalar and \mathbf{u} is any vector in W , then $k\mathbf{u}$ is in W .

This Theorem as stated is completely correct. But a student who uses these two conditions as the defining properties of a subspace will eventually come upon a potential subspace that is empty: it will satisfy (a) and (b) without being a subspace. The phrase “one or more vectors” hidden in the hypothesis makes the result correct, but hides a property that is really essential.

Presumably errors are not included by design. Being able to make corrections between editions of a book is impossible, and new editions must be infrequent because of financial constraints; it is trivial to make corrections if the source is online.

1.2 Motivation #2: Breaking the Linear Structure of a Book

Looking at other textbooks is really no help. All seem to have problems such as those given above to a greater or lesser extent. At least part of the problem lies not with the authors, but rather with the nature of the textbook. In particular a textbook must

- assume a certain background and level of innate ability for the student, and give argument in detail consistent with that assumption,
- have a manageable finite length (although some linear algebra textbooks are now over 700 pages long!),
- be essentially linear (each chapter uses results proven in earlier chapters).

1.3 Motivation #3: It's Easier to be an Editor than an Author/Editor

Writing a book is a huge task. Revising it for later editions is almost as bad. However, writing a small section of a book is pretty easy, and it's usually not too difficult to get volunteers to undertake such a task. Similarly, the quality and the levels of difficulty of the exercises in a book is an important (and often underestimated) aspect of the presentation. Getting volunteers to contribute a modest number of exercises is not too difficult. Of course, as with any open project, there must be an editor (dictator) in charge to ensure that all contributions to the project are consistent.

2 Why a wiki

The motivations given in the last section drive the choice of software towards a wiki[3], and many of the problems mentioned in the previous section are solved easily by this choice. The greatest asset is that the data does not have a linear structure as a book does but, rather, is organized as a tree. There are several implications of this structure:

- The same material may be examined from different viewpoints. The most straightforward one may be given in a page with links to deeper or more subtle arguments.
- Forward references are natural; many times it is both expedient and useful to defer a proof of a mathematical concept until a better understanding of the underlying structures has been acquired. By custom, a book will give the proof immediately after its statement (or not at all!).
- Tangential but interesting topics can be explored via a link without interrupting the general flow of the main text.
- More repetition is possible: a theorem may have several similar parts with subtly different proofs. There is simply not enough room in a book to repeat similar proofs (a standard trick is to move some proofs to the exercises), but it is not a problem (and even encouraged) with separate links for comparable material.

There are other advantages with a wiki:

- The use of colour is quite expensive for a book; it costs nothing in a wiki.
- Photographs are quite expensive for a book; it costs nothing in a wiki.
- Putting a copy of a textbook as a file on a student's computer as, say, a PDF file is (copyright) problematic; it is straightforward with a simple wiki extension.
- The inclusion of accurate graphics (e.g., two dimensional or three dimensional plots) is also straightforward.
- The use of animations is easy.

Students taking a first-year university course are usually still in their teens. This means, for the current crop of students, that the home computer has always been commonplace, and that any software more than ten years old has been around forever. In particular, they have never known a time when computers didn't come with a browser that could access Wikipedia. Wikis for them are like radios for us: we know that there was a time when they didn't exist, but we can't really imagine it. Just as we can't really appreciate the awe that must have been felt by an adult who, for the first time, heard sound coming out of thin air, our first-year students feel that it is no big deal to have an enormous database like Wikipedia available from their bedrooms on a 24/7 basis.

This is, in fact, a wonderful opportunity. The student comes with a skill set that can be very useful. I have, by intention, made my wiki look much like Wikipedia. By doing this, I don't have to tell students where to look for help, or where the print button is: they just do the right thing.

As another example, a theorem may have several parts with each dependent on the proofs of the previous ones or with all of them logically equivalent. In either case, the part may be stated with a link labelled "Proof" next to it. No further explanation is necessary; the student knows exactly what to do.

3 The first steps

To test the efficacy of the wiki in the classroom, I volunteered (!) to teach a large-section, one-term linear algebra course over several terms. The students in this course have varied levels of ability and ambition. Some just want to be through with their math requirements while others know that they want an honours degree in mathematics. The content of the course is not atypical: Solving systems of linear equations, Gauss-Jordan reduction, some two and three dimensional geometry, and an introduction to vector spaces.

The initial wiki pages were actually expanded lecture notes. Students were encouraged to read them and print them out in advance of the lectures. They were also encouraged to find errors (and indeed they did!). The basic structure for the course and the problems assigned came from the textbook as usual. In a certain sense, this was an anti-wiki: the material was edited by one person and the advantage of the cooperative nature of a wiki was just thrown away. Arguably, the first step in building a quality implementation is to run a simpler prototype and find out what actually happens (goes wrong, in particular), and so that is the approach that was used. Feedback from the students could then be used as an aid for deciding the best way to proceed. The reactions of students were gauged in two ways:

- The access count of each page was monitored. In this way it was possible to see which pages were actually used.
- Comments on the wiki were invited as part of our standard student evaluation questionnaire given at the end of the course.

A second initiative concerning the wiki implementation involved the method of presentation of the mathematics on the web pages. Three approaches were used:

- The "default" approach: using software on the server to generate and cache png files of the mathematics used on the page.
- MathJax, which uses JavaScript within the browser to create the mathematics for the page on-the-fly.
- Translate a math snippet into MathML, which is then displayed using the abilities of the browser.

All three of these have been used as part of the initial tests of the wiki with varying results which are described in more detail in the Subsection 5.1.

4 Initial results

The response from students to the questionnaires has been overwhelming positive. Over 90% are strongly supportive of the idea of having a wiki for the presentation of course materials. They had some additional interesting ideas within the latest questionnaire (May 5, 2011):

Very fond of the homemade Wikipedia. Made finding information very simple.

The wiki was actually very awesome—the only thing missing was some kind of interactive quiz that tested knowledge.

It would be great if the wiki showed where we were in the course at any given time.

Wonderful substitute for textbook and great was to save students a few bucks, and take advantage of new and different technologies.

There were a few really negative responses, but unfortunately they didn't go into detail. It might have been really helpful to know the reasons for their feelings.

There were also some qualified endorsements.

The wiki is helpful but there are a few math errors that need some editing.

The wiki pages took a long time in downloading and sometimes didn't work. The note printouts were missing the symbol θ sometimes so there would be a blank space left.

Many responses indicated that the student preferred a dynamic wiki to a standard textbook. How could anyone be surprised by this? Nonetheless, some questions to ask are:

- Will the wiki actually increase the level of competence of the students? For example, giving solutions to (at least some of the) exercises is certainly desirable. If these solutions are only a click away, will the temptation to look before actually working on the exercises be too great to resist?
- Will the tail wag the dog? The goal is to help the student learn the mathematics, and not to entertain them. A sense of proportion is necessary.
- Mathematics must be seen as more than a collection of algorithms. Does the wiki format help or hinder this process?

5 Implications of Display Methods

5.1 Inserting Mathematics within the Text on the Page

Several methods were used to display mathematics on the wiki pages. Each came with assets and liabilities:

- The default (`texvc`) method [2]: the material designated as mathematics is sent through \LaTeX to create a `dvi` file. This file is sent through `dvi.png` to create a `png` file. The `png` file is then inserted into the page as a graphic image. This method works quite well: it is very fast and produces accurate results. However, the results are bitmapped, and, as such, can not be changed when the font sizes in the display are changed (by the user or otherwise). There are usually no problems with pixellation, but the potential for problems is there for complicated constructions (the rendering of subscript *i* versus subscript *iota*, for example).

- The MathJax method: MathJax is a javascript application [1] which may be used to render mathematics on wiki pages. It produces beautiful results that may be enlarged when the font size of the surrounding text is changed. It can also be used with some of the special L^AT_EX packages (for example, amsmath) available for mathematicians. However, each time a new page is accessed, it is run through MathJax and the time required to render a complicated page may be lengthy. In addition, when jumping to a specific location on a new page, the position may be lost as the page is rendered.
- The MathML method: MathJax can translate the material designated as mathematics into MathML code. An appropriate browser is necessary, of course. When such a browser is used, the mathematics is rendered much more quickly than with MathJax using javascript. There still is some delay on complicated pages. The problem of the loss of position when the page is rendered is also a problem with this method.

In short, the rendering of the mathematics on wiki pages is still a work in progress. The solutions currently being used are still pretty new. I expect them to improve a lot in the next few years.

It is interesting to note the students' solution to the display problems. If they couldn't read the screen, then just pushed the print button and looked at the resulting pdf file. It looked fine most of the time.

5.2 Multimedia displays

The use of multimedia software is, of course, one the most exciting prospects within a wiki site. The dynamic content at this point consists of animated gifs. These have been produced using different approaches:

- 1. Use T_EX to make a dvi file of each image.
- 2. Make a cropped PostScript file of each image using dvips.
- 3. Convert the PostScript file to a gif.
- 4. Stitch the gifs together to make an animated gif.
- Use a symbolic manipulator (Maple in this case) to make an animation and save it as an animated gif.
- Make a shell script to generate PostScript files, and stitch them together in an animated gif.

6 Next steps

The work done so far is just one step in a larger project. The next goal is a wiki covering undergraduate linear algebra. It will be structured in such a way to allow the material to be used in different ways for different courses.

Any list of topics is necessarily idiosyncratic. Initially the list of topics covered will include:

- Vector spaces over \mathbb{R} , \mathbb{C} , finite fields and general fields.
- Subspaces. Linear independence, generating (spanning) sets, bases. Dimension. Infinite dimensional vector spaces. Direct sums. Every vector space has a basis (application of Zorn's Lemma or some other form of AC).
- Matrices. Matrix algebra. Inverses of matrices. Singular matrices. Determinants. Rank of matrices. Equivalent conditions for nonsingularity. Row space, column space, null space, rank of matrices and their interrelations.

- The theory of systems of linear equations. Elementary matrices. Gaussian Elimination. The reduced row echelon form. Solution space of a homogeneous system, particular and general solutions of an arbitrary system.
- Linear transformations. Representation by matrices in the finite dimensional case. Fundamental subspaces associated with a transformation (kernel, range). Fundamental subspaces associated with a matrix and their connection to linear transformations. Dimension theorem.
- Linear operators on finite and infinite dimensional spaces.
- Linear functionals. Dual spaces. Duals of finite dimensional spaces; duals of infinite dimensional spaces.
- Eigenvalues and eigenvectors of a linear operator; of a matrix. Characteristic equation of a matrix. The minimal polynomial. Diagonalization of matrices (operators) and applications. Invariant subspaces. Cayley-Hamilton theorem.
- Inner product spaces. Cauchy-Schwarz theorem. Orthogonal complements. The Gram-Schmidt algorithm. Adjoint of a linear operator. Unitary, self-adjoint, normal and orthogonal operators and their matrices. Diagonalization and self-adjoint operators.
- Jordan canonical form.
- Bilinear forms and their representations. Quadratic forms. Positive definite and positive semidefinite bilinear forms.

Each topic will have a page with a table of contents (automatically generated) that includes the list of theorems, and will include separate links to proofs when appropriate, and many exercises of varying difficulty. Solutions to all problems will be available.

It will also become a collaborative project. A working group has been established and is looking at finishing the next stage in a 12–18 month time period.

The beauty of the wiki is that the material may be expanded or changed easily and on a collaborative basis. It also keeps track of the dependencies between pages, an invaluable tool when choosing which topics to cover. Alternative approaches are easily implemented.

The approach so far has been incremental—this matches the philosophy of the wiki perfectly. Further expansion after completing the next stage is anticipated.

References

- [1] MathJax: <http://www.mathjax.org/>
- [2] MediaWiki with texvc: <http://www.mediawiki.org/wiki/TeXvc>
- [3] The experimental wiki can be accessed at <http://wikitest.cs.umanitoba.ca/mathwiki/index.php/Math1300:MainPage>

The Web of Mathematical Models*: A Schema-based, Wiki-like, Interactive Platform

Thomas Grundmann, Jean-Marie Gaillourdet, Karsten Schmidt, Arnd Poetzsch-Heffter, Stefan Deßloch

Department of Computer Science
University of Kaiserslautern, Germany
{thg,jmg,kschmidt,poetzsch,dessloch}@cs.uni-kl.de

Martin Memmel

Knowledge Management Department, DFKI GmbH, Trippstadter Str. 122
D-67663 Kaiserslautern, Germany and
University of Kaiserslautern, Germany
martin.memmel@dfki.de

Abstract

In science and engineering mathematical models are increasingly important to describe natural phenomena and design artifacts. Our goal is to make the notion of “mathematical models” more explicit and precise as well as to build up knowledge repositories for searching, exploring, combining, and sharing models. With the *Web of Mathematical Models, WoM*, we provide a platform to host such models on the Web. Models follow an explicit, content-related schema.

1 Introduction

In science and engineering mathematical models are increasingly important to describe natural phenomena and design artifacts. Not least because of the power of modern software and computer technology, which allows for better analysis, visualization, and comprehension of natural phenomena and designed artifacts. Based on Eykhoff’s definition, in which a mathematical model is “a representation of the essential aspects of an existing system (or a system to be constructed) which presents knowledge of that system in usable form” [6], we distinguish three kinds of models:

- *Descriptive models* that explain the essential aspects of existing systems such as physical, sociological, or economical systems;
- *Constructive models* that describe systems to be constructed as part of engineering tasks or processes and
- *Abstract models* that are used for modeling a certain class of phenomena, but are not (yet) applied to a specific system. Examples are special classes of differential equations or labeled transition systems.

Our goal is to make the notion of “mathematical models” more explicit and precise as well as to build up knowledge repositories for searching, exploring, combining, and sharing such models. With the *Web of Mathematical Models, WoM*, we provide a platform to host such models on the Web. Its mission is to help improving the accessibility, usability, precision, tool support, classification, and comparability of mathematical models, and thus, *WoM* provides a foundation for future computer-guided design flows and an intelligent engineering support for standardized, composable, and computer-processable models. To achieve our goals, we had to answer the following two central questions:

*The project *WoM* is supported by the Rhineland-Palatinate research center $(CM)^2$ – Center for Mathematical and Computational Modelling.

1. How to represent mathematical models?
2. How to manage and present them in an open model platform?

To answer the first question, we established a small initial user community of applied mathematicians and identified the following requirements:

- A model should have an *informal description* that introduces the model and explains which phenomena or artifacts are modelled. It may be supported by graphical or video visualizations of the model.
- A *mathematical description* characterizes the model in terms of its mathematical properties. It explains the parameters of the model. In principle, it should be expressible in a formal language.
- *Software support* allows simulating the model with different parameter settings. The software should be realized in a component-based way such that models can be composed.
- *Linking and metadata* relate the model to other models and related documents, and provide support for classification and structured search.

These requirements are fulfilled by an XML-based schema. This schema does not only structure the model definition, but also relates it to visualizations – graphics or videos – and (interactive) simulation programs. It supports two kinds of simulations:

- *pre-fabricated simulations* that are embedded into the Web user interface and create – based on a set of parameters – a result, e.g., a picture or a movie, and
- *open simulation programs* that are embedded into an interactive environment, which allows the user to experiment with and to explore the model in interactive sessions.

WoM provides the possibility to integrate remotely usable simulation programs without in-depth knowledge of Web technologies. Since we anticipate authors to be mathematicians or engineers, but not Web developers, we expect that WoM simplifies the publishing of models for them.

To answer question two, we identified four requirements which enable us to store, relate, query, search, use, and compose models:

- *Community support*: Construction, collection, classification, and linking of models is only possible with support from a user community. Accordingly, the platform should encourage participation, and foster the development of a self-sustainable community.
- *Model construction*: The platform should support the schema-conformal construction and modification of models. It should combine Wiki functionality with an offline editing possibility. In particular, models need a declarative and platform independent representation that can be down- and uploaded.
- *Web accessibility*: Models and all their functionality should be accessible and usable on the Web. In particular, simulations should be possible without download and shareable between users.
- *Evolvability*: To stepwise realize our vision, many changes of the schema for model representation and ontologies for classifications have to be managed in the future. In particular, the existing model representations have to evolve together with the schema and ontologies in a consistent way. This is only achievable with mechanical support by the platform.

Addressing these last four requirements, we developed a web-based model platform¹. It is centered around a repository for our schema-based model representation. We expect the schema to be crucial for evolvability.

Web accessibility is ensured by the integration of the ALOE system into the *WoM* infrastructure. ALOE² is a web-based and generic social resource sharing platform developed at the Knowledge Management group of DFKI³. ALOE allows contributing, sharing, organizing, and accessing arbitrary types of digital resources such as text documents, music, or video files. Users are able either to upload resources or to reference them using URLs. Furthermore, the platform offers common user management features and Web 2.0 interaction possibilities like tagging, rating, and commenting on resources.

This paper focuses on the schema used for the representation of mathematical models (Section 2). It extends the overview paper [7]. In Section 3, we discuss related work. Section 4 contains the conclusion.

2 A Schema for Models and their Relations

Our approach is built around a structured representation of models. Based on an XML-based schema, we started to build up a model repository. Having an explicit, content-related schema distinguishes the approach of *WoM* from classical Wiki platforms and enables stronger computer-support for model use and management. The schema is called XModel Schema and is internally defined by an XML Schema. \LaTeX serves as the default input language – in particular for formulas. Thus, we provide a \LaTeX implementation of the XModel Schema. Future versions may also support other input languages like MathML. In this section we introduce the current state of the XModel Schema.

The *XModel Schema* defines an XML representation of a model. The components of an XModel – an instance of the XModel Schema – are shown in Figure 1 and described in the following paragraphs.

Describing a Model. Besides a *title*, an XModel essentially consists of an informal description and a formal description. The *informal description* is basically a textual representation of the model. It is a compound of *paragraphs*, *lists*, (simple) *tables*, and *images*. In addition, elements to *emphasize* content and for *referencing* are supported as well as *mathematical expressions*. Although they are permitted, these expressions do only have a descriptive character and are not associated with any special role.

In contrast, the *formal description* defines a model using mathematical expressions, which have special meaning. They are used to define the model's set of *input parameters* I , *output parameters* O , and *miscellaneous parameters* M . The latter may serve as constants or other variables. Each parameter is defined by a *name*, a *domain/type*, and a short textual *description*. These parameter blocks are followed by the *model's definition*, which describes how the model is implemented using the beforehand defined parameters. Therefore, this consists of text content similar to the informal description's one. To (re)use these parameters, special elements to refer to a parameter's name, domain/type, and description are added (*parameterName*, *parameterType*, *parameterDescription*). But the essential part of the definition block is a set of *model equations/formulas* R that define the relationships between the parameters, such that for each $o \in O$ at least one equation/formula $r(I' \cup M') \in R$ exists with $I' \subseteq I$ and $M' \subseteq M$. R may also contain equations/formulas that allow to bind some miscellaneous parameters $m \in M$ for further use in other equations.

¹A snapshot of a development prototype is available here: <http://angren.cs.uni-kl.de/WoMstatic/>. Use *womguest* as username and password to login.

²<http://aloe-project.de>

³German Research Center for Artificial Intelligence

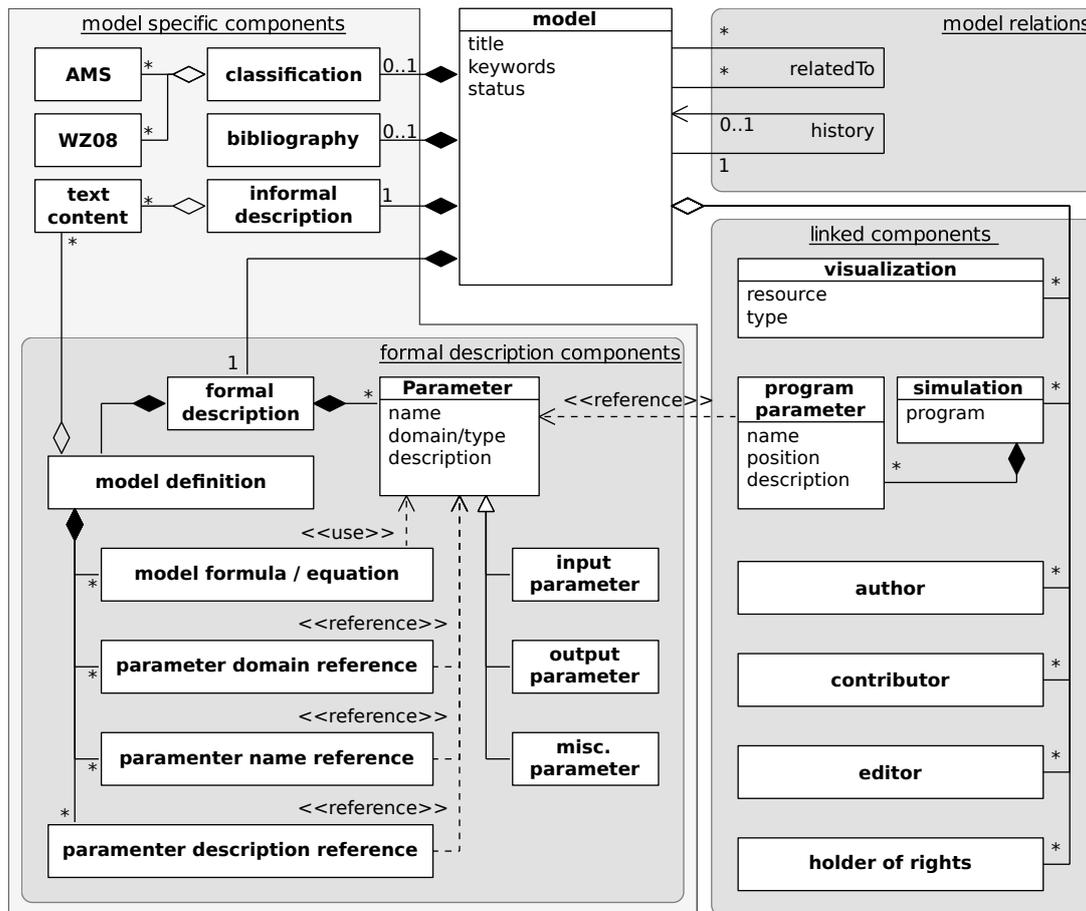


Figure 1: The XModel Schema

Visualizations and Simulations. In addition, a model may contain visualizations and simulations. The *visualizations* can range from simple images to complex videos. *Simulations* are provided by software packages that allow for online experimentation with the model. They usually take some input parameters and – in the case of pre-fabricated simulations – return an image or video as result. These *program parameters* should correspond to the (input) parameters of the formal model. Therefore, the formal description’s parameters can be referenced by the program parameters. Thus, the program parameter’s role (name, description, and possibly domain/type) can be easily inferred. In addition to these correlated parameters, a simulation may also have parameters to control particular aspects of the simulations. Currently Matlab- and Sage-based simulations are supported. The Sage framework [16] provides a web-based Python interpreter and a very rich library of mathematical algorithms and data structures.

While the previously described way to use simulations by referencing an external program resource, for instance a Matlab or Sage file, is already supported by the *WoM*, we are currently developing *model-embedded simulations*, which are (Sage-based) programs that allow to define a simulation directly inside the XModel. Therefore, the XModel Schema allows code snippets that are parts of the formal description’s parameters and equations/formulas. An accordingly extended schema is depicted in Figure 2. It shows the extension of the parameters and equations with the new element *simulation code* that encapsulates the additional code. Using IDs, each simulation code element is associated with the simulation to which it belongs. Because the order of the snippets in the description may be different to their order in the program, additionally, each simulation code element has to be numbered. For code, which does not

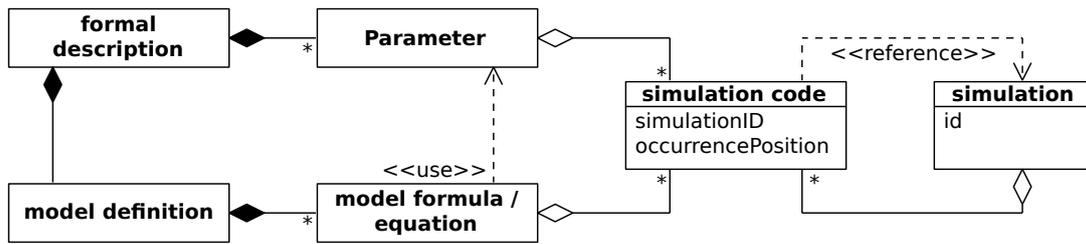


Figure 2: XModel Schema with integrated simulations

directly belong to a formal description’s component, such snippets can be defined within the body of the corresponding simulation element.

Metadata. The XModel Schema also includes metadata such as the model’s *authors*, *contributors*, *holders of rights*, and *editors* as well as *keywords*, the model’s *status* that can be stable, experimental, or checked, and a *bibliography*, whose structure is based on the BibTeX syntax. Using a *classification*, a model can be related to standard classifications. Currently, *WoM* supports the AMS 1991 Mathematical Subject Classification [1] and the WZ08 - Classification of Economic Activities [4]. A model may also be *related to* other models in the *WoM*, which is a symmetric relation.

The XModel Schema is designed in a way that allows to evolve the models over time. For example, models may be classified according to a new classification ontology or may be extended with new simulations and visualizations. Of course, changes in other parts of the model are possible as well. Consequently, a model may have different versions and provides access to its *history*.

Processing an XModel Document. As mentioned in the beginning of this section, we currently provide a \LaTeX document class, which roughly describes the syntax of XModel. The \LaTeX source is translated into an XModel document. Because of that, LaTeXXML [12] in conjunction with BibTeXXML [2] are used to translate the \LaTeX document into an intermediate XML document, which is finally transformed into an XModel document using XSLT. Based on a valid XModel representation, further processing operations like a transformation into an XHTML representation or back into a \LaTeX document are possible. The entire processing chain is depicted in Figure 3.

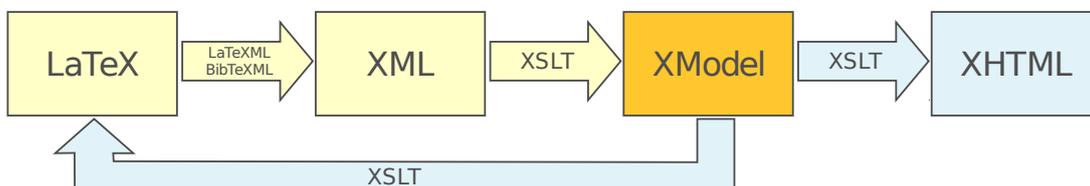


Figure 3: The processing chain from \LaTeX to XHTML

In Figure 4, the result of such an additional transformation is shown. It exemplarily illustrates the XHTML representation of a model for the “Horizontal Shot”. Furthermore, it shows the integration of a pre-fabricated Matlab simulation in section 4. *Simulations*. A separate link to a corresponding Sage simulation is shown at the end of that section. Figure 5 gives an impression of the Sage interface.

Horizontal Shot

[stable] Visualization Simulation

Created by [D. Poetzsch-Heffter](#) [WWW] @ ITWM
 Keywords: shot; mechanics; physics; school

1. Informal Description

The horizontal shot describes the ballistic trajectory of a masspoint (i.e. it moves without friction) on the planet earth. This model is one of the basic models of the mechanics theory founded by Isaac Newton.

2. Formal Description

The basic situation of the model is that the masspoint is s_{y_0} metres above the ground and is shot with an initial speed named v_x parallel to the ground.

$$v_y(t) = gt$$

$$s_y(t) = \frac{1}{2}gt^2 + s_{y_0}$$

$$v_{tot}(t) = \sqrt{v_x^2 + v_y(t)^2}$$

$$\alpha(t) = \arctan\left(\frac{v_y(t)}{v_x}\right)$$

As you can see, v_y , s_y , v_{tot} and α depend on the input parameter t . By using the following equation it is also possible to determine these values using the way parallel to the ground s_x as input parameter:

$$t = \frac{s_x}{v_x}$$

2.1. Input Parameters

- s_{y_0} The initial distance of the masspoint to the ground; $s_{y_0} \in \mathbb{R}_0^+$
- v_x The x-direction (parallel to the ground) part of the speed; $v_x \in \mathbb{R}^+$
- t The elapsed time; $t \in \mathbb{R}_0^+$
- s_x The way in x-direction (parallel to the ground); $s_x \in \mathbb{R}_0^+$

2.2. Output Parameters

- v_y The speed towards the ground; $v_y \in \mathbb{R}^+$
- s_y The distance to the ground; $s_y \in \mathbb{R}$
- v_{tot} The absolute value of the masspoint's speed; $v_{tot} \in \mathbb{R}^+$
- α The angle between the masspoint's direction of movement and the ground; $\alpha \in [0 - 90]$

2.3. Miscellaneous Parameters

- g The gravitational acceleration; $g = -9.81 \frac{m}{s^2}$

3. Visualizations



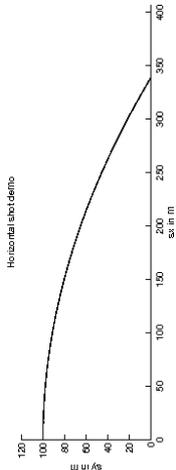
A ballistic trajectory shown via a water jet

4. Simulations

The movie of the following simulation is a realtime movie. This means the duration of the movie is exactly the time the masspoint would need to reach the ground in reality.

The initial distance of the masspoint to the ground s_{y_0} [m]

The x-direction (parallel to the ground) part of the speed v_x [m/s]



I have prepared an interactive environment for plotting diagrams of horizontal shots (shown under this passage). Use function `plot_trajectory(sy0, vx)` to plot the curves.
[Open SageMath simulation window](#)

5. Classification

(AMS)70B05 — Kinematics of a particle

6. References

- Isaac Newton: *Philosophiæ Naturalis Principia Mathematica* (1687)

7. Related models

[Vertical shot](#) [General shot](#)

Figure 4: Web representation of the “Horizontal Shot” XModel

Sage Interactive

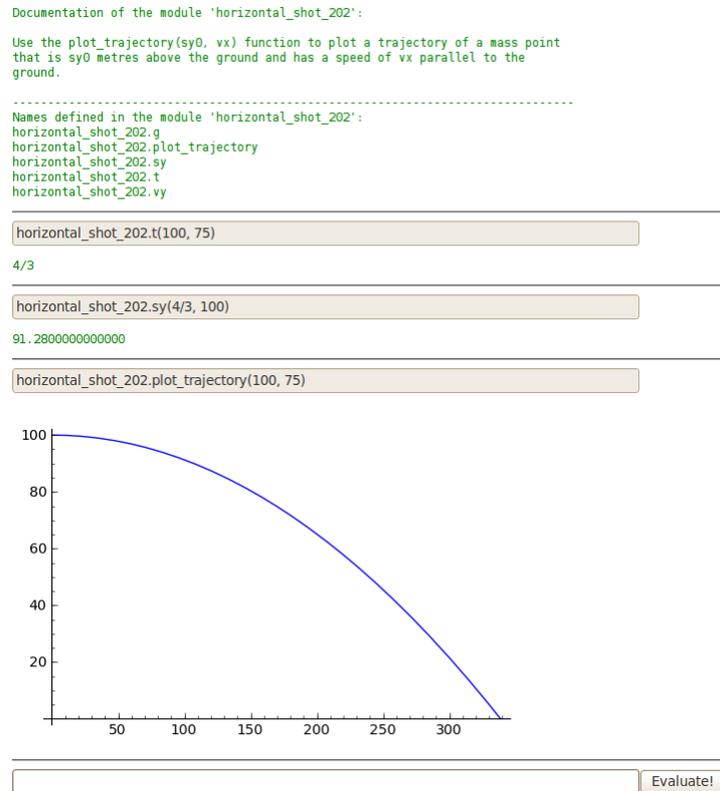


Figure 5: The Sage simulation provided by the Sage module which is part of the model “Horizontal Shot”.

3 Related Work

Several communities work on different aspects to support math in the Web. Some of them focus on providing visual aids for mathematical algorithms, like the proprietary platform available under [17], which is based on Mathematica/WolframAlpha. *Demonstrations*, which are mainly interactive, are based on a CDF⁴ file that can only be processed by proprietary tools. For various demonstrations unstructured text comments and author information are given. Demonstrations may be referenced to “related” demonstrations, but that is not necessarily a symmetric relation. It also misses any community features, model structuring, and alternative visualization capabilities.

Another approach are math-related Wikis, which are based on several freely available software projects. A wide range of them use the OMDoc [9] format⁵. It concentrates on the representation of mathematical equations, their transformation, and referencing. Others use the OpenMath and MathML standards [3]. Sample projects are [10] (outdated) and [15] – “The Encyclopedia Sponsored by Statistics and Probability Societies”. While they are providing AMS classifications, user communities, and \LaTeX -based integration of new content, they lack a clear model structure. For instance plain \LaTeX documents and visual support are limited to ordinary images. Another prototype *JOBAD* (JavaScript API for OMDoc-based Active Documents) [8] concentrates on the integration of other (ad-hoc called) Web

⁴Computable Document Format

⁵OMDoc is an inter lingua for mathematical communications.

Services. They are linked via keywords or explicit user input. This project's main goals are on-the-fly conversion of units, and document visualization style. Neither interactive models nor structured models are provided.

The PlanetMath project [14] collects – similar to a math encyclopedia – all kinds of math-related and \LaTeX -sourced documents and makes them available as HTML pages. The downloadable intermediate XML format does not provide any clear structuring or support for model linkage, besides keywords, which are automatically extracted from the text. While community features like comments and history are supported, neither visual assistance, except for images, nor interactive playgrounds are provided.

Platforms like ActiveMath [11] are capable of communicating with computer algebra systems or formalizing mathematical expressions in order to annotate or simply present them in the Web. Some of them provide learning platforms that allow flash programs and applets to be embedded. Special platforms (see e.g. [13]) are specialized in proof languages and proof checker capabilities for all math-related expressions. But none of them provide any community features nor do they support the notion of mathematical models and their relations.

While the *WoM* and the OKSIMO project (formerly known as Planet Earth Simulator [5]) have in common that they want to model day-to-day problems and make the partially interactive models available on the Web, the OKSIMO project depends on a propriety Fcl input language, i.e., a visual programming language, to submit new models. In contrast to *WoM*, OKSIMO lacks a structured model repository.

In summary, the described approaches have different focuses, but usually share some technical aspects. For instance, JOBAD uses a similar model repository approach, namely TnTBase – a database assembled from Subversion and Berkeley DB XML. Except platforms aiming at formalizing mathematical expressions, all (web-oriented) platforms support \LaTeX -based inputs without any pre-defined additional structure. A distinguishing feature of the *WoM* approach is that it supports its models/entries by an explicit schema.

4 Conclusion

We introduced the Web of Models, a platform for storing, searching, exploring, and sharing mathematical models. In this paper, we focused on the structured representation of a model as an instance of the XModel Schema. It defines the scope of *WoM* and, besides others, allows to classify models as well as relate them to each other. It standardizes the model's structure and thus improves model consistency. It allows to integrate simulations whether as references to an already existing program or directly inside the model's description via model embedded simulations. With the help of such simulations the user can explore and experiment with the models. Created from a \LaTeX document an XModel instance can be transformed into different representations like XHTML. Finally, XModel is the cornerstone of *WoM*, of the model platform built around, and for all available transformation and processing options. Moreover, it supports any future evolution steps by automated processing capabilities.

References

- [1] American Mathematical Society. <http://www.ams.org/>.
- [2] BibTeXML. <http://www.arakhne.org/bib2ml/>.
- [3] O. Caprotti and D. Carlisle. OpenMath and MathML: semantic markup for mathematics. *Crossroads*, 6:11–14, November 1999.
- [4] Statistisches Bundesamt Deutschland. <http://www.destatis.de/>.

- [5] Gerd Doeben-Henisch. The planet earth simulator project - a case study in computational semiotics. In *AFRICON, 2004. 7th AFRICON Conference in Africa*, volume 1, pages 417–422, September 2004. <http://oksimo.inm.de/>.
- [6] Pieter Eykhoff. *System Identification: Parameter and State Estimation*. Wiley, 1974.
- [7] Jean-Marie Gaillourdet, Thomas Grundmann, Martin Memmel, Karsten Schmidt, Arnd Poetzsch-Heffter, and Stefan Deßloch. WoM: An Open Interactive Platform for Describing, Exploring, and Sharing Mathematical Models. In A. König et al., editor, *Knowledge-Based and Intelligent Information and Engineering Systems, 15th International Conference, KES2011*, Lecture Notes in Computer Science, 2011. to appear.
- [8] Jana Giceva, Christoph Lange, and Florian Rabe. Integrating web services into active mathematical documents. In Jacques Carette, Lucas Dixon, Claudio Coen, and Stephen Watt, editors, *Intelligent Computer Mathematics*, volume 5625 of *LNCS*, pages 279–293. Springer Berlin / Heidelberg, 2009.
- [9] Michael Kohlhase. OMDoc: Open mathematical documents. In Fred Vries, Graham Attwell, Raymond Elferink, and Alexandra Tödt, editors, *Open Source for Education*, pages 137–143. Open Universiteit Nederland, November 2005.
- [10] MathWeb. <http://www.mathweb.org>.
- [11] Erica Melis, Giorgi Gogvadze, Paul Libbrecht, and Carsten Ullrich. ActiveMath – a learning platform with semantic web features, 2009. <http://www.activemath.org>.
- [12] Bruce Miller. LaTeXML. <http://dlmf.nist.gov/LaTeXML>.
- [13] A. Naumowicz and A. Kornilowicz. A brief overview of Mizar. In S. Berghofer and et al., editors, *TPHOLS 2009*, 2009. LNCS 5674.
- [14] PlanetMath. <http://planetmath.org>.
- [15] StatProb. The encyclopedia sponsored by statistics and probability societies. <http://statprob.com>.
- [16] W.A. Stein et al. *Sage Mathematics Software*. The Sage Development Team, 2011. <http://www.sagemath.org>.
- [17] Wolfram. Wolfram demonstrations project. <http://demonstrations.wolfram.com>.

Wiki Authoring and Semantics of Mathematical Document Structure

Hiraku Kuroda* and Takao Namiki
Department of Mathematics, Hokkaido University,
060-0810 Sapporo, Japan

Abstract

We are developing a CMS including document authoring feature based on wiki to publish structural mathematical documents on the Web. Using this system, users can write documents including mathematical expressions written in \LaTeX notation and explicitly stated characteristic structures of mathematical articles such as definitions, theorems, and proofs. Documents input to the system is published on the Web as not only XHTML files to be browsed but also XML files complying with NLM-DTD, which is used to exchange articles electronically. Not only single wiki page document, users can build a document which consist of more than one pages and is described its structure semantically by the system. In order to do this, we also propose an application of OAI-ORE and RDF vocabularies to describe structures of documents consisting of several resources.

1 Introduction

Today, many documents are published on the Web. The term “documents” here includes articles of news or blog, wiki pages, journal articles, and any other web pages. These documents are published as HTML or XHTML to be browsed, or as PDF or PS file to be printed out. Sometimes one document is published as several formats.

Some documents consist of several resources. One of examples is a document including a graphic image. Here, we assume that body text of the document is written in a HTML file and the image is a JPG file. On the Web, each of them is independent resource and given unique URI. When URI of the image is put on `src` attribute of an `img` element in the HTML file, we should treat the document as not just referencing but including or embedding the image. In this case, this document is an aggregation of two resources that are HTML file of body text and JPG file of graphical image.

Sometimes documents include not only graphic images but also whole of other (more small) documents. In general, this is called *transclusion*. HTML does not have this transclusion feature in itself, but MediaWiki, for example, has *templates* feature to extract content of other wiki pages to the page [3]. In this case, the document is an aggregation of resources that are body text written in wiki markup and other documents which are indicated in the document to be included.

Furthermore, we sometimes build a document by integrating several documents. In this case, not only a large document is just split into several documents, but each of documents is independent and has their own URI, and they can be referenced directly. In general, parts, chapters, or sections of a document are able to be independent documents. MathML specification by W3C [8] is an example of such documents. This document consists of one overview, eight sections, and eleven appendices. These divisions are independent web pages and have their own URIs. Finer divisions of a document may be independent documents according to structure or characteristics of a document. For example, definitions, theorems, proofs, and expressions in mathematical documents may be independent documents.

Open Archives Initiative Object Reuse and Exchange is standards to describe and exchange aggregations of web resources [11]. In the User Guide of OAI-ORE [13], journal articles are described as

*hiraku@math.sci.hokudai.ac.jp

Listing 1: A document including a theorem written in Wiki

```

At first , we give Taylor 's theorem and proof of it .

[[theorem id="taylor_theorem" title="Taylor 's theorem"
Let  $f$  be a function which is defined on the interval  $(a,b)$  and suppose the  $n$ th
derivative  $f^{(n)}$  exists on  $(a,b)$ . Then for all  $x$  and  $x_0$  in  $(a,b)$ ,


$$R_n(x) = \frac{f^{(n)}(y)}{n!}(x-x_0)^n$$


with  $y$  strictly between  $x$  and  $x_0$  ( $y$  depends on the choice of  $x$ ).  $R_n(x)$  is the
 $n$ th remainder of the Taylor series for  $f(x)$ .
]]

(Original text of the theorem is http://planetmath.org/encyclopedia/TaylorTheorem.html ,
retrieved at 2011.05.08)

```

aggregations of representation files such as PDF or PS. In this article, on the other hand, we propose describing documents as aggregations of constituting resources and relating the documents with their representations apart from describing aggregations.

With a background like that, we are developing a content management system **Matherial**, which manages and publishes mathematical documents and other resources. One of the purposes is developing a system which assists to write documents consisting of several resources, and publishes as web pages with appropriate metadata to describe its structure and publishes as XML files complying with NLM-DTD for further reusing.

Matherial provides authoring assistant feature based on wiki engine. Users of the system can write a wiki page including chapters, sections, mathematical statements, and expressions, or they can write some of them as independent wiki pages and integrate them into one document and publish it. Relationships between documents and included resources, and between documents and wiki pages representing them, are modeled as aggregations of OAI-ORE, and they are described in XHTML representation of documents by RDFa. Matherial can output documents into not only one or more XHTML pages, but also XML files complying with NLM-DTD. Therefore, other systems supporting NLM-DTD are able to re-use documents by Matherial.

The paper is organized as follows: In section 2, we present an example of mathematical structural documents on Matherial. In section 3, we propose an application of OAI-ORE and RDF vocabulary to describe structural documents on Matherial and more generally on the Web. Finally, section 4 concludes the paper.

2 Mathematical Contents Management System

2.1 Wiki-based Authoring

One of major features of **Matherial**, developed in this study, is assistant authoring mathematical documents. This is based on so-called “Wiki Engine”, so users can write documents by simple markup notation and publish them on the Web. They can put mathematical expressions written in \LaTeX notation into texts, and our own **MathML library** [7] converts them to MathML [8].

The most simple type of documents created on Matherial is one consisting of a wiki page. When users need to write mathematical text structures, such as definitions, theorems, and proofs, using functional markup for them, they can expressly provide that segments have such property.

For example, Listing 1 is a document written in wiki markup including a theorem. When users write theorems in their document directly like this, URIs of theorems are hash URIs, appending

Listing 2: A document import other resources

```

We begin with Taylor's theorem and its proof.

[[import wiki/TaylorTheorem]]

[[import wiki/ProofOfTaylorTheorem]]

For a function  $f(x)$ ,  $f$  is Taylor expandable when  $\lim_{n \rightarrow \infty} R_n(x) = 0$  where  $R$  is
remainder term of [[wiki/TaylorTheorem|the theorem]], and we have bellow.

[[import wiki/TaylorExpansion]]

Even if complex function  $f(z)$  is not holomorphic at a point  $c$ , if  $f$  is holomorphic in an
annulus around  $c$ , we get Laurent series bellow,

$$f(z) = \sum_{n=-\infty}^{\infty} a_n (z-c)^n$$

where

$$a_n = \frac{1}{2\pi i} \oint_{\gamma} \frac{f(z) dz}{(z-c)^{n+1}}$$

and  $\gamma$  is a closed curve in the annulus([ref annulus]).

[[figure file/AnnulusOfLaurent id=annulus]]

This is extension of [[TaylorExpansion]] for functions which are not holomorphic.

```

their IDs as fragment to URI of the document. In this case, assuming a URI of a document is <http://mw2011.matherial.org/wiki/Taylor>, a URI of a theorem itself in the document is http://mw2011.matherial.org/wiki/Taylor#taylor_theorem.

For important definitions, theorems, and proofs, considering we discuss about them or reuse them from other documents, they should be independent documents and referenced by their own URI. On wiki of Matherial, users can set type of page, for example set that page is a theorem, the system treats the document by the wiki page as if it is described a theorem. In this case, URI of the theorem is URI of the document by wiki page. Detail about URIs and relationships of documents and wiki pages in Matherial are illustrated in section 3.

With Matherial, users can write documents importing and extracting mathematical statements which have been created as independent document. Moreover users can put images which are managed in Matherial into documents in the same way, and they can use descriptions of images which were input when images were upload to the system instead of writing new descriptions in the page. Listing 2 is a document importing statements which are already published and going on to describe a statement following them. In that example, an image referenced in the text will be imported with its description.

Moreover, aggregating these documents as sections, chapters, or parts, users can build a new document. In Matherial, users input enumeration of sub documents with metadata of the document such as title, author's information, and so on into form to build the document. Detail of semantic structure of documents which consist of several resources is described at section 3.

2.2 Output Documents

Matherial output documents as XML files. XML schemas of output XML files are XHTML [18] to read directly by web browsers, and NLM-DTD [9] to exchange articles electrically.

For XHTML files, metadata are described as RDF graph [14] and embedded by RDFa [15]. Metadata written into XHTML are metadata of the document itself such as title, authors' information, and time and date when the document was created and update, and structure information about relationships between the document and other resources. For example, relationships between resources for a document about the Laurent series shown previously is illustrated at Fig.1. The web page of this document browsed is

Listing 3: A part of an NLM-DTD XML version of a document

```

<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//NLM//DTD Journal Archiving and Interchange DTD v3.0 20080202//EN"
"archivearticle3.dtd">
<article>
  <front>
    <article-meta>
      <title-group><article-title>Laurent Series</article-title></title-group>
      <contrib-group>
        <contrib>
          <name><surname>Kuroda</surname><given-names>Hiraku</given-names></name>
        </contrib>
      </contrib-group>
      <pub-date><day>29</day><month>5</month><year>2011</year></pub-date>
      <self-uri xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://mw2011.matherial.org/LaurentSeries/en"/>
    </article-meta>
  </front>
  <body>
    <p>We begin with Taylor's theorem and its proof.</p>
    <statement-content-type="theorem">
      <title>Taylor Theorem</title>
      <p>Let <inline-formula><math xmlns="http://www.w3.org/1998/Math/MathML" display="inline"><mi>
*snip*
      <attrib>
        <uri xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://mw2011.matherial.org/LaurentSeries/en/pr/TaylorTheorem"/>
      </attrib>
    </statement>
    <statement-content-type="proof">
      <title>Proof of Taylor Theorem</title>
*snip*
      <attrib>
        <uri xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://mw2011.matherial.org/LaurentSeries/en/pr/ProofOfTaylorTheorem"/>
        <uri xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://mw2011.matherial.org/TaylorTheorem" xlink:role="http://matherial.org/term/proofOf"/>
      </attrib>
    </statement>
*snip
and <inline-formula><math xmlns="http://www.w3.org/1998/Math/MathML" display="inline"><mi>&#
x3B3;</mi></math><inline-formula> is a closed curve in the annulus(<fig.<xref rid="annulus">
annulus</xref>).</p>
  <fig position="float" id="annulus">
    <graphic xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://mw2011.matherial.org/files/2011/05/10/0/file"/>
    <attrib>
      <uri xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="http://mw2011.matherial.org/LaurentSeries/en/pr/file/2011/05/10/0"/>
    </attrib>
    <caption>
      <title>Annulus for Laurent Series</title>
      <p>Annulus for Laurent Series is shown.</p>
    </caption>
  </fig>
  <p/>
  <p>This is extension of <ext-link xmlns:xlink="http://www.w3.org/1999/xlink" ext-link-type="uri" xlink:href="http://mw2011.matherial.org/TaylorExpansion">TaylorExpansion</ext-link> for functions which are not holomorphic.</p>
</body>
</article>

```

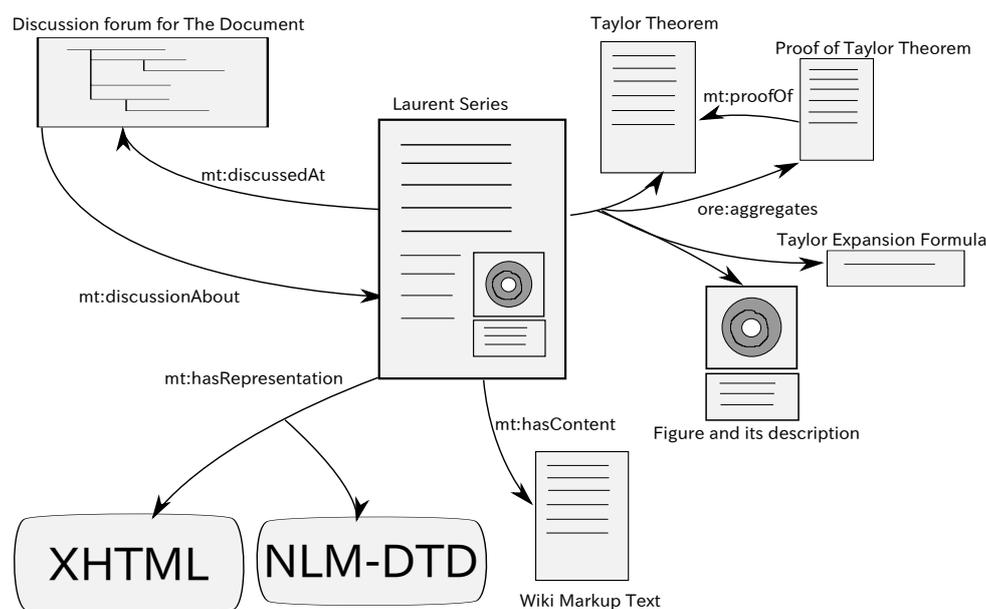


Figure 1: RDF graph of a document consisting of several resources

An RDF graph describing relationships between the document, included resources, representations and a discussion forum about the document is shown.

shown at Fig.2.

For XML files complying with Archiving and Interchange Tag Set [10] of NLM-DTD, URIs of included resources are written at `xlink:href` attribute of `uri` element in elements which included resources are extracted to. Mathematical statements are extracted into `statement` element, and type of statements are explicitly shown at `content-type` attributes. Listing 3 is a part of an XML file complying with NLM-DTD of the document shown previously. Mathematical statements are written into `statement` elements and their URLs are into `statement/attrib/uri` elements. Relationship between a theorem and its proof is shown at `statement/attrib/uri` element of the proof. Imported image and its description are extracted at `fig` element and it is referenced using `xref` element. You can get the whole of the XML file from <http://mw2011.matherial.org/LaurentSeries/en/nlm>.

3 Document structure and its metadata

The documents authoring feature of Matherial is based on wiki engine. Users write texts by wiki markup of Matherial. The most simple document is one consist of only body text but not any other resources. Matherial converts an input wiki markup text to XML files complying with XHTML and NLM-DTD, and publish them on the Web. Wiki source files, XHTML files, and XML files should be given different URI, and a URI of each files is different from the URI of the document itself (we call a URI for a document itself *platonic form URI*) [16].

In Matherial, users can write documents which include other resources, too. This does not means that documents just reference other resources. As `\includegraphics` command or `\input` command of \LaTeX , users can embed images or extract contents of other documents into the document. Matherial generates XHTML files which include contents of other documents and is embedded images by `img` elements, and generates XML files of NLM-DTD which include other documents and is embedded

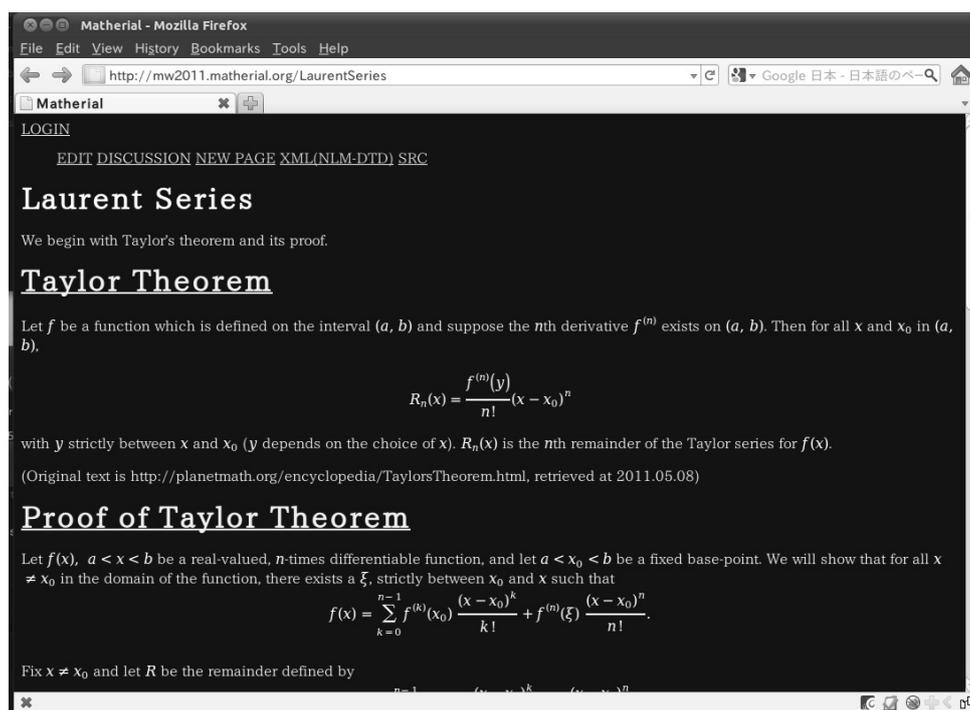


Figure 2: A web page of a document importing other documents

A web page of a document importing other documents is shown. Imported documents are extracted. Mathematical expressions written in \LaTeX notation are converted to MathML and rendered by web browser. URI of this page is <http://mw2011.matherial.org/LaurentSeries/en/html>. You can also browse this page by using platonic form URI of this document, <http://mw2011.matherial.org/LaurentSeries/>.

mt	http://www.matherial.org/terms/
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
ore	http://www.openarchives.org/ore/terms/

Table 1: Name spaces and Prefixes

URI of prefix mt is namespace for experimental vocabulary in this study. URI of prefix rdf is namespace for basic vocabulary of RDF [14]. URI of ore is namespace for vocabulary of OAI-ORE [11].

images by graphic elements, from wiki sources written like that.

This document structure on Matherial is described as an RDF graph whose nodes are platonic form URI of the document, URIs of resources included in the document, URIs of representations of the document, and so on (Fig.1). Matherial describes this structure by Resource map of Open Archives Initiative Object Reuse and Exchange (OAI-ORE) [11].

The structure of a document which consists of several resources and which is represented by several representations could be applied to not only documents in Matherial but general documents on the Web. In following subsection 3.1, we will introduce OAI-ORE to describe Aggregations of resources, then in subsection 3.2, we will show a model of the document structure and metadata schema to describe the structure. RDF namespaces and its prefixes used in this article are shown at table 1.

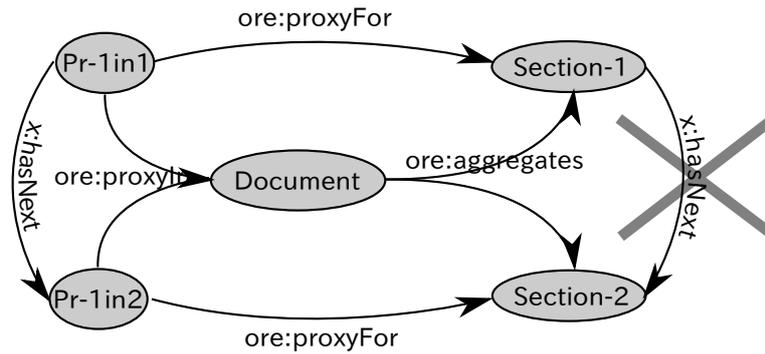


Figure 3: Proxy of OAI-ORE

Section-1 and Section-2 are Aggregated Resources in an aggregation Document. Two proxies Pr-1in1 and Pr-1in2 are Proxies for two Resources in the Aggregation. Order relation `x:hasNext` should not set between two Aggregated Resources directly.

3.1 Revisiting OAI-ORE

Open Archives Initiative Object Reuse and Exchange (OAI-ORE) provides a model to describe *aggregations of resources*. In OAI-ORE, a resource which consists of one or more resources is called an **Aggregation**. A resource which is a member of an Aggregation is called an **Aggregated Resource**. An aggregation is defined as a conceptual construct, so it does not have a representation. Therefore, information about an Aggregation should be described by other resources different from the Aggregation. Such a resource which describes an Aggregation is called a **Resource Map**. For example, relationships between an Aggregation and Aggregated Resources are described in a Resource Map for the Aggregation.

OAI-ORE provides a mechanism named **Proxy** to relate Aggregated Resources with properties which are given only in the Aggregation. In an Aggregation A-1, for example, we assume that two Aggregated Resources AR-1 and AR-2 have order $AR-1 \rightarrow AR-2$. Moreover, we assume that they have different order $AR-2 \rightarrow AR-1$ in another Aggregation A-2. In this case, if we describe directly the order relationship between AR-1 and AR-2 in AR-1, it conflict to relationship in AR-2. This is because we describe relationship which is available only in AR-1 independently of AR-1. To resolve this problem, we use **Proxy** resources which act as *Resources in the Aggregation* to describe relationships available only in the Aggregation between Aggregated Resources and other resources. For example, relationships between AR-1 and AR-2 in A-1 is described using their Proxies like fig.3.

For more detail of OAI-ORE, see [11].

3.2 Structure of Document including Resources

3.2.1 Document and its Members

In this study, a **Document** is an Aggregation consisting of one or more resources. A resource which is a constituent of a Document is called a Member of Document, or simply a **Member**. An image embedded into a Document is familiar example of Member. A Member of a Document could be another Document different from the aggregating Document. In other words, aggregating several Documents, we can build a new Document. Relationships between a Document and its Members are described by RDF triples whose predicates are `ore:aggregates`.

Document Content. A Document could have a resource as one of Members which is described the content of the Document itself. We call such a Member a **Document Content**. A content of a Document is body text of the Document. When a Document includes other resources, indications to include them are written into the content. One example of Document Content is a HTML file, which is a source file of a web page. We can write not only marked-up body text, but also indications to embed images into the page using `img` elements. In this case, the Document consists of a HTML file as Document Content and images indicated. When we browse this document, web browser get a HTML file from a server, recognize it, get other resources to embed into the page, and display the completed web page. For another example, text files written in wiki markup for any wiki engines. They are described body text and embedding indications different notation from HTML. The system may convert it to a HTML, or create a PDF file which contains image files. Relationship between a Document and a Document Content is described by a RDF triple whose predicate is `mt:hasContent`, which is sub property of `ore:aggregates`.

Order of Members. When Document has the Document Content, positioning of other Members is described in the Document Content. On the other hand, when Document is simple Aggregation of resources and does not have a Document Content, we may want to describe positioning or order of Members. Furthermore, we may want to give Members complicated and non-linear order relationships such as tree structure. In this article, we propose `mt:hasNext` predicate to describe order relationships of Members in the Document. This property takes URIs of Proxies of Members for subjects and objects of triples to describe linear or complicated order relationships of Members of the Document (fig.4)

Type of Members. We may want to give Members any role in a Document. In an “article” document, for example, the first Member is abstract of the article, following some Members are Sections of the article, and the last Member is References of the article.

`mt:partType` predicate is to describe these roles of Members in a Document. This property takes URIs of Proxies of Members for subjects, and URIs of sub-classes of `mt:PartType` which represent roles of Members in Documents. Sub-classes of `mt:PartType` are `mt:Preface`, `mt:Abstract`, `mt:TableOfContents`, `mt:Part`, `mt:Chapter`, `mt:Section`, `mt:Acknowledgment`, `mt:Appendix`, `mt:References`, and `mt:Index`.

3.2.2 Mathematical Element

Mathematical documents may contain distinctive elements, such as mathematical expressions (especially *display math style*), definitions, theorems, proofs. When we write a mathematical document, preparing these elements as independent resources and including them in the Document as Members, we can reference these import elements individually and reuse them.

When we create a Document containing mathematical elements, we can show type of the Document explicitly using sub-classes of `mt:MathematicalObject`. Sub classes of `mt:MathematicalObject` are `mt:Expression`, `mt:Definition`, `mt:Theorem`, and `mt:Proof`. `mt:Theorem` has more detailed sub classes, that are `mt:Lemma`, `mt:Corollary`, and `mt:Proposition`.

A resource of type `mt:Proof` describing mathematical proof should show explicitly which theorem is proved. `mt:proofOf` is predicate for RDF triples to relate theorems and its proofs. This property takes URIs of resources of type `mt:Proof` for subjects and URIs of resources of type `mt:Theorem` or its sub classes for objects (fig.4).

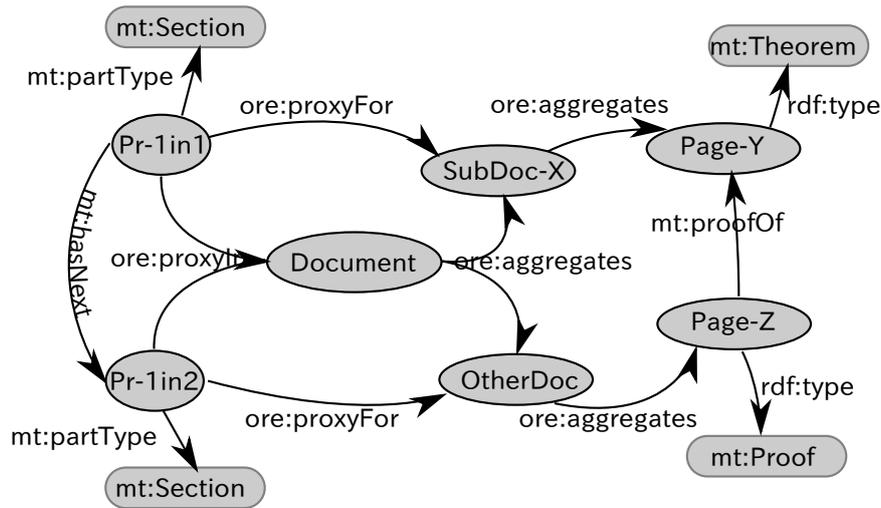


Figure 4: Structure of mathematical Document

An example of Documents which has several Documents as Members is shown. Members include independent theorem and its proof. Relationships of them are described.

3.3 Document and its Representations

In this article, Documents are Aggregations of ORE, so Documents are abstract resources and do not have entities. Therefore, to browse Documents, Documents should be related with other resources which Documents are serialized in any formats which we can browse. A resource which is serialized from a Document and has URI different from URI of the Document is called a **Representation** of the Document. When a Document is related to a Representation of the Document, We say *a Document has a Representation*. a Document Content may be one of Representations of the Document. Some Representations include all Members of Document any way like PDF files, other Representations include only indications and references to other Members like HTML files. Relationships between Documents and its Representations are described by RDF triples using `mt:hasRepresentation` predicate (fig. 1).

4 Conclusion and Discussion

In this article, we introduced a CMS developed for authoring and publishing mathematical documents, and we proposed background semantics to describe structures of documents consisting of several resources. Using the system, we can publish not only small documents but also large documents consisting of several resources by writing in easy mark-up. Structures of documents consisting of several resources are described as RDF graphs based on Resource map of OAI-ORE, and they will be reused by Semantic Web Technologies.

Some applications of OAI-ORE are describe an article as an aggregation of OAI-ORE. The FORESITE [2] project developed a toolkit to describe metadata of articles from JSTOR by using OAI-ORE. In the project, each issue of journals is an Aggregation of articles, and each article is an Aggregation of individual page images and a PDF-formatted version of the entire article [1]. The ICE-TheOREM project [17] provides thesis authoring and publishing systems. In this project, each thesis is an Aggregation of sections and PDF, DOC, and ODT version of the article, and each section is an Aggregation of PDF, DOC, and ODT version of the section. These applications treats each article as an Aggregation

of parts of it and its Representations. On the other hand, in this article, we describe a Document as an Aggregation of parts of it, and we use another property for relationships between a Document and its Representations.

The OMDoc format is a content markup scheme for mathematical documents [4]. This format is designed for the Mathematical Knowledge Base. SWiM is a semantic wiki for Mathematical Knowledge Management using OMDoc and OpenMath [5][6]. While these are aimed at building the Mathematical Knowledge Base, the Matherial is aimed at publishing mathematical documents by using simple notation for authoring and only presentation markups for outputting. OMDoc also provides a document ontology [12]. RDF classes i.e. Definition, Theorem (and so on), Proof, and Formula and RDF properties i.e. proves and provedBy are defined, but a class for general mathematical expression is not defined. These classes of OMDoc ontology are subclass of MathKnowledgeItem. However, documents of Matherial are not expressed in OMDoc. So we use classes in mt namespace instead of OMDoc ontology.

References

- [1] H. V. de Sompel, C. Lagoze, M. L. Nelson, S. Warner, R. Sanderson, and P. Johnston. Adding eScience Assets to the Data Web. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web*, June 2009.
- [2] FORESITE. <http://foresite.cheshire3.org/>.
- [3] Help:Templates MediaWiki. <http://www.mediawiki.org/wiki/Help:Templates>.
- [4] M. Kohlhase. *OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]*, volume 4180 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [5] C. Lange. Mathematical Semantic Markup in a Wiki: the Roles of Symbols and Notations. In *Proceedings of the 3rd Semantic Wiki Workshop (SemWiki 2008) at the 5th European Semantic Web Conference (ESWC 2008)*, June 2008.
- [6] C. Lange. SWiM – A Semantic Wiki for Mathematical Knowledge Management. In *The Semantic Web: Research and Applications 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings*, June 2008.
- [7] MathML Library. http://rubygems.org/gems/math_ml.
- [8] Mathematical Markup Language (MathML) Version 2.0 (Second Edition). <http://www.w3.org/TR/MathML2/>.
- [9] Journal Archiving and Interchange Tag Suite. <http://dtd.nlm.nih.gov/>.
- [10] Archiving and Interchange Tag Set. <http://dtd.nlm.nih.gov/archiving/>.
- [11] ORE Specifications and User Guides - Table of Contents. <http://www.openarchives.org/ore/1.0/toc>.
- [12] OMDoc Document Ontology. <http://kwarc.info/projects/docOnto/omdoc.html>.
- [13] ORE User Guide - Primer. <http://www.openarchives.org/ore/1.0/primer>.
- [14] RDF - Semantic Web Standards. <http://www.w3.org/RDF/>.
- [15] RDFa in XHTML: Syntax and Processing. <http://www.w3.org/TR/rdfa-syntax/>.
- [16] L. Richardson and S. Ruby. *Restful Web Services*. O'Reilly Media, 2007.
- [17] P. Sefton, J. Downing, and N. Day. ICE-Theorem - End to end semantically aware eResearch infrastructure for theses. *Journal of Digital Information*, 11(1), Jan. 2010.
- [18] XHTML 1.1 - Module-based XHTML - Second Edition. <http://www.w3.org/TR/xhtml11/>.

Ideas for a MathWiki Editor

Sebastian Reichelt
SebastianR@gmx.de

Abstract

We present some functional and non-functional requirements and wishes for a web-based editor for formalized mathematics, in particular for use in the MathWiki project at RU Nijmegen [13]. We discuss possible implementation alternatives, and argue for a holistic design of the entire wiki with editor features in mind.

1 Introduction

Since the invention of proof assistants, researchers have argued for a library of mathematics formalized in a machine-readable format; this goal is stated and explained in the QED manifesto [1], for example. Such a library would have a vast amount of use cases from the verification of complicated proofs (as in the Flyspeck project [4]), to computer algebra systems with strong correctness guarantees, and to learning environments for students to become familiar with mathematical proofs and check their results [17]. Most importantly, it could serve as a uniform repository for present and future mathematical theories, to ensure that no mathematical developments become “lost” in the ever-growing body of results.

Formal math differs from other variants of mathematics done on a computer in that definitions, statements, and proofs are built from a limited set of basic principles, so that the computer can be said to actually “understand” the contents of the library (to the extent possible). This enables to a degree of correctness and homogeneity that cannot be achieved any other way.

However, the amount of work necessary to build such a library is prohibitive for any single person or project [16]. One possible solution is to organize its development in a collaborative fashion, using wiki-like technology [13]. In contrast to regular wikis, only meaningful formal definitions and correct proofs can be entered. Still, a combination with informal content is possible and beneficial, potentially bringing together formerly separate user communities.

Compared to desktop-based proof assistant IDEs, the wiki approach trades performance and simplicity for ease of use and availability. The major benefits of a web-based solution are the lack of client-side installation and the ability to work directly on a single consistent library. Performance is not expected to be a large problem as long as the number of contributors is low. If the required server-side computation becomes too expensive in the future, a hybrid desktop/internet solution (i.e., client software accessing a remote library) may become a better alternative. At the moment, we favor a fully web-based solution because of its potential to attract more contributors in the first place.

A major caveat, however, is that formalization is not only time-consuming but also rather difficult. In particular, the learning curve is currently too steep to appeal to a significant number of novice users. Moreover, formal math is still closer to the source code of a computer program than to informal math [17]. For this reason, the features of a wiki editor strongly affect the potential user base: The more guidance and readily accessible information the editor provides, the easier and quicker the input of formalized mathematics will become, all other things being equal.

In addition, since new users first face the obstacle of having to learn about already existing formal content, it is even more important for the static (non-editing) part of the wiki to provide as many cues as the accompanying editor; one way to achieve this is to use the same rendering mechanisms in viewing and editing mode wherever possible. Since websites showing appropriately post-processed formal mathematics already exist (e.g. `isarmathlib.org`), the actual challenge lies in bringing the same features to an editor. Present systems such as the current MathWiki editor [13] or `wikiproofs.org` offer essentially

a raw text editor, in contrast to feature-rich desktop IDEs like Proof General [2] or CoqIDE [12]. The ProofWeb system [6] contains an editor modeled after such IDEs, but it is specific to Coq at the moment and does not offer many of the features we envision. Especially, incorporating ProofWeb would imply that viewing and editing rely on entirely different code bases, so features would have to be implemented twice.

To develop an editor for the MathWiki project [13], we would especially like to leverage some of our experience gained from developing the HLM proof assistant [8, 9]: Its defining characteristic is a graphical user interface that is tightly integrated with the verifier component, so that many useful features can be implemented directly on top of the formal data structures. However, the intention of MathWiki is to provide access to a number of different existing proof assistants, such as Mizar [10] and Coq [12]. This is a compromise: Ideally, we would like to specify the formal content just once, using a rich user interface with HLM-like features and a general interchange format such as OMDoc [7], and use the resulting data to generate formalization for different proof assistants. However, since the conversion of formal mathematics between different proof assistants is problematic [18] (at the moment, at least), supporting different proof assistants in parallel seems to be the safer route to take.

This paper presents the result of a first investigation into the possibilities of implementing auxiliary features (for example those known from proof assistant IDEs or from HLM) in a MathWiki editor, on top of different provers.

2 Requirements

As indicated in the introduction, an editor for a mathematical wiki must be both easy to use and flexible with respect to the underlying prover technology. In this regard, different modes of interaction of existing proof assistants present a special challenge. Even just considering technological differences and ignoring mathematical foundations, there are actually several dimensions along which provers differ:

- The most well-understood dimension concerns the proof language, which can be either declarative or procedural [19]. Roughly speaking, declarative proofs consist mostly of lists of statements that are proved to hold, along with hints that guide the prover towards the verification of these statements. Procedural proofs contain commands (or “tactics”) that tell the prover exactly how to proceed (in contrast to the hints in declarative proofs, which merely need to contain enough information to prove the statement in question). Since these commands fully describe the proof, intermediate results are usually not included in the proof script, and can only be obtained by “replaying” the proof in the prover. (The Proviola tool [11] aims to address this shortcoming.) One requirement for a MathWiki editor is that procedural proofs should be just as easy to edit as declarative proofs. In other words, the editor needs to provide additional information to the user in order to make procedural proofs readable.
- A related but separate dimension is whether the prover maintains some internal state in addition to the input text. In most procedural provers, every line of input constitutes a state change. This state influences the information that is shown to the user and is necessary to understand procedural proofs. Coq, in particular, also has an “undo” feature to revert the last step [5]; this enables an editor like CoqIDE [12] to maintain a movable cursor indicating the portion of the input text that has been sent to the prover. If the proof language is declarative, no additional state is necessary; Mizar [10] is an example of a stateless prover. However, declarative languages have been developed for stateful proof assistants as well. Also, HLM [9] can be described as procedural but stateless: Its user input comes from context menus instead of text, and the contents of these menus

contain the information that would normally depend on the prover state (but in this case merely depends on the the location or context of each menu, i.e. on the library contents).

- The graphical input method in HLM presents another challenge: It would seem that a text editor would be a basic ingredient of a generic MathWiki editor, but HLM proofs cannot feasibly be edited as text. On the other hand, an additional graphical interface would be helpful even if the primary input is in text form.
- Finally, a web-based front end is usually inherently asynchronous, while most existing software operates in a synchronous fashion (recent Isabelle developments being a notable exception [15]). There are two reasons for the asynchronous nature of websites: the potentially large latency of all operations that require client/server communication, and the standard mode of operation of text editor controls in browsers. In a declarative and stateless scenario, asynchronous verification is unproblematic: Whenever the input text changes, the prover can re-verify it and show the results to the user when they are available. Stateful provers are more difficult to connect to an asynchronous front end because the intended, actual, and observed state can diverge quickly. Finally, HLM constitutes a special case again: It is inherently synchronous because possible inputs are determined by menu contents that change after every operation which modifies the library.

Some compromises are necessary to accommodate all flavors of proof assistants, even in principle. In addition, the editor must integrate well with the rest of the wiki. A number of features (most of which are available in HLM, for example) would be desirable both in the wiki and its editor. We will briefly characterize their value according to the methodology of Cognitive Dimensions (CD) [3]. Of the four types of user activity mentioned in CD literature, incrementation, transcription, and exploratory design seem especially important at the current stage, whereas modification (of existing formal content) will most likely remain the job of a few experts for the foreseeable future. In addition, the ability to understand and browse existing data is vital, even though it is not classified as a user activity in the CD sense.

- The most obvious enhancement in a web-based viewer and editor is the use of (automatically generated) hyperlinks to navigate to referenced definitions and theorems. In terms of Cognitive Dimensions, such links improve the “visibility,” or accessibility, of referenced objects, enabling exploration and modification. Since placing links in a text editor may be difficult to impossible, links may need to be shown separately in editing mode.
- When the user moves the mouse over a clickable link, an abbreviated version of the linked item can be displayed as a tooltip. This feature has the potential to greatly increase usability of a mathematical wiki because it reduces the number of pages that need to be opened in order to understand a given item. The general user-friendliness of tooltips stems from their non-disruptiveness: They typically disappear whenever they would stand in the way. However, since mathematical definitions are often complex, the size of the tooltip area can become a problem, especially since tooltips will typically appear at locations where they hide relevant content of the current page. Thus, an even less disruptive alternative would be a dedicated area on the page instead of a floating tooltip. In CD terms, temporarily showing the contents of a referenced item corresponds to the “juxtaposition” of that item with the one the user is currently viewing or editing. This helps the user understand the contents of the current item more quickly, and can also prevent errors due to incorrect definitions or theorem statements.
- Definitions, theorems, and proofs should be rendered in a visually pleasant form. One important ingredient is the use of common mathematical symbols; for example, the author of a definition

could specify a custom symbol that represents the defined object at all places where it is used. Ideally, it should be possible to reproduce the usual mathematical notation even in cases where that involves more than a single symbol. In HLM, the notation for a definition in the library can be specified as a two-dimensional “layout” that includes placement of arguments and is augmented by further information such as rules for parentheses. The inclusion of arguments in the notation is possible because library entries cannot be referenced as mathematical entities; like “functors” and “predicates” in Mizar they are always referenced with specific arguments. In type-theoretical proof assistants, such definitions yield functions in the mathematical sense, which can be referenced on their own. In this case, the notation feature can only be reproduced approximately.

A user-defined notation for mathematical objects is a “secondary notation” according to CD, as it provides visual hints beyond the raw formal content, aiding in transcription and modification (assuming it is actually available in the editor, not just in the viewer). In this context, it especially reduces the “hard mental operation” of deciphering formal mathematics, by relating it to known informal math. This is desirable for all possible user activities.

- Although outside of the scope of this paper, we propose a tree or a tree-like menu of all definitions and theorems for navigation in the wiki, which should be available in all situations. As HLM shows, such a tree is much easier to browse if items are shown in their custom notation, and previews of items are shown as tooltips prior to opening them.
- The text editor should include syntax highlighting, at least for basic keywords (which is the prime example of a “secondary notation”).

3 Design Alternatives

Different approaches are possible depending on the importance of each requirement. In the current MathWiki implementation, the wiki and editor are entirely separate. Since the editor is a raw text editor at the moment, non-essential features like hyperlinks are available only for finished formalizations, after they have been submitted and verified. The simplest approach would be to enhance the text editor with as many features as possible, for example automatic asynchronous verification and highlighting of errors.

Although this approach is compelling especially because of its incremental nature, it bears two significant problems: First, most of the information that the user needs in addition to the raw text has to be displayed separately, and updated through a complex client/server protocol. Second, such an individual piece of software easily becomes more and more detached from the rest of the wiki as more features are added to it. For example, to accommodate stateful provers with procedural proof languages, there needs to be a display of the current state, but this display is then unavailable on the main page even though it would be equally important there.

In other words, such a design would be feasible but rather short-sighted: Over time, similar features would be desired on both wiki pages and editor pages, but most of the features described in section 2 would need to be implemented twice. In addition, the differences between provers could lead to separate editor implementations, requiring further duplication of features.

At the other end of the spectrum, there is the possibility of displaying and editing everything at a higher level, hiding the underlying textual representation. If the high-level representation is fully equivalent to its textual counterpart, it can be used equally for viewing and editing formal content, and all conceptual differences between proof assistants can be concealed by this abstraction layer.

This is very similar to the approach taken by the HLM proof assistant, although HLM goes one step further by omitting the textual representation entirely. The idea of HLM is that everything (including definitions, statements, and proofs) is displayed in a natural mathematical style, and input happens via

menus which contain pre-rendered versions of the corresponding result. For example, if the goal of a proof is a universally quantified statement $\forall x \in S : P(x)$, the menu item corresponding to universal generalization will simply show “Let $x \in S$.” In terms of Cognitive Dimensions, this reduces the “hard mental operation” of having to figure out the command for universal generalization (or even just realizing that universal generalization is the correct next step).

The existence of HLM shows that this method of proof input is viable but requires complex dialogs to input parameters of proof steps. One should also keep in mind that the HLM logic is designed especially to facilitate menu-based input; it is difficult to imagine how the same mechanism could be used as the primary or sole input method for an existing proof assistant.

However, a middle ground exists as well. The basic idea is to provide both a textual and a high-level representation side-by-side (or just the high-level representation if HLM is used as the underlying system). The additional high-level view provides all of the desired features such as hyperlinks, tooltips, custom notation, etc., but only limited editing facilities. It is shared between the main page and the editor page.

While this might seem like an obvious solution, the actual difficulty lies in the connection between these two views. With a declarative, stateless, and asynchronous system underneath, the high-level view can simply be updated at regular intervals. For example, if the Mizar system is used, an existing Mizar-to-XML translation [14] appears suitable as an intermediate representation from which a readable version of the document can be computed. With HLM, there is no text input, and all editing happens synchronously in the high-level view. However, procedural stateful provers present a challenge because the user expects to see the current state, and because a change at one position in the input text tends to break all commands beyond that position.

Our proposed solution is to limit the high-level view to the commands that have already been verified, and to merge the state display into it. In the case of proving $\forall x \in S : P(x)$ as above, at the beginning the high-level view simply contains this goal statement. After the user submits the appropriate command for universal generalization, a new line is added to the high-level view, showing “Let $x \in S$. Then $P(x)$.” to indicate the current hypothesis and goal.

In general, the contents of the high-level view are computed from all of the prover states after sending each command to the prover, up to the current state, rather than from the raw input text. Thus, no additional parsing of user input is necessary, and the display can be enhanced with all useful information that can be obtained from the prover.

If the user has to trigger every state change manually (for example using “up,” “down,” and “go to cursor” buttons as in existing proof assistant IDEs), the connection between both views becomes very loose, in contrast to the automatic updating in declarative mode. This problem becomes worse in a web front end because custom keyboard shortcuts are usually not available. However, because of the importance of the state display, the user presumably needs full control over the position that separates the verified and unverified parts.

Thus, we suggest that the input cursor be used to determine this position, which is equivalent to an automatic “go to cursor” operation whenever the cursor position changes. In particular, whenever the user finishes entering a command, that command is automatically sent to the prover. Besides requiring less keystrokes, a special advantage is that the verified part of the text does not need to be locked: If the user moves the cursor into this part in order to edit it, the prover will be instructed to backtrack to this position anyway. The lack of locking makes the editor “less synchronous,” mitigating one of the differences between provers.

This feature can be regarded as an extended variant of the “electric terminator mode” available in Proof General [2]. The difference is that Proof General only reacts to the input of specific characters terminating the pieces of text that can be sent to the prover on their own; it does not change the prover state every time the user moves the cursor or presses backspace to remove a “terminator” character.

Although the elimination of all explicit navigation is a rather radical change from the user’s point of view, first experiments with an implementation in CoqIDE look encouraging: The lack of interruptions from regular text input actually tends to make proof input somewhat smoother.

4 Conclusion and Future Work

We have presented requirements and design alternatives for an editor that is integrated into a mathematical wiki. The desire to support several proof assistants with different interaction styles, and to present formal content in a more high-level form than raw text, requires a compromise between a text editor and a structural view or editor. We have argued for a side-by-side presentation both in the editor and in the wiki itself, and described how a text editor can be connected to a high-level view, depending on the interaction mode of the underlying prover.

The next step will be to implement, within the MathWiki framework, the proposed method of interacting with provers. A particularly interesting question is how well the automatic “go to cursor” feature works together with asynchronous updating of the input text, and how much a delayed display of the current prover state (due to network latency) affects usability.

Many thanks go to Josef Urban for his support and very helpful discussions, and to the anonymous reviewers for their detailed comments (including a pointer to the concept of Cognitive Dimensions).

References

- [1] The QED manifesto. In *Proceedings of the 12th International Conference on Automated Deduction, CADE-12*, pages 238–251. Springer-Verlag, 1994.
- [2] David Aspinall and Thomas Kleymann. *Proof General user manual*. <http://proofgeneral.inf.ed.ac.uk/userman>.
- [3] Thomas Green and Alan Blackwell. Cognitive dimensions of information artefacts: a tutorial. In *BCS HCI Conference*, 1998.
- [4] Thomas C. Hales. Flyspeck: A blueprint of the formal proof of the Kepler conjecture. <http://flyspeck.googlecode.com/files/flypaper.pdf>.
- [5] G. Huet, G. Kahn, and Ch. Paulin-Mohring. *The Coq Proof Assistant – A tutorial*, April 2004. <http://coq.inria.fr/getting-started>.
- [6] Cezary Kaliszyk. Web interfaces for proof assistants. In S. Autexier and C. Benzmüller, editors, *Proc. of the Workshop on User Interfaces for Theorem Provers (UITP’06)*, volume 174[2] of *ENTCS*, pages 49–61, 2007.
- [7] Michael Kohlhase. *OMDOC – An open markup format for mathematical documents [Version 1.2]*. Number 4180 in *LNAI*. Springer Verlag, 2006.
- [8] Sebastian Reichelt. The HLM proof assistant (website). <http://hlm.sourceforge.net/>.
- [9] Sebastian Reichelt. Treating sets as types in a proof assistant for ordinary mathematics. (Unpublished note accompanying presentation at TYPES’10.) <http://hlm.sourceforge.net/types.pdf>, 2010.
- [10] Piotr Rudnicki. An overview of the MIZAR project. In *Types for Proofs and Programs*, pages 311–332, 1992.
- [11] Carst Tankink, Herman Geuvers, James McKinna, and Freek Wiedijk. Proviola: a tool for proof re-animation. In *Proceedings of the 10th ASIC and 9th MKM international conference, and 17th Calculemus conference on Intelligent computer mathematics*, AISC’10/MKM’10/Calculemus’10, pages 440–454. Springer-Verlag, 2010.
- [12] The Coq Development Team. *The Coq Proof Assistant Reference Manual*. <http://coq.inria.fr/refman/>.
- [13] Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. A wiki for Mizar: motivation, considerations, and initial prototype. In *Proceedings of the 10th ASIC and 9th MKM international conference, and*

- 17th Calculemus conference on Intelligent computer mathematics, AISC'10/MKM'10/Calculemus'10*, pages 455–469. Springer-Verlag, 2010.
- [14] Josef Urban and Grzegorz Bancerek. Presenting and explaining Mizar. *Electron. Notes Theor. Comput. Sci.*, 174:63–74, May 2007.
- [15] Makarius Wenzel. Asynchronous proof processing with Isabelle/Scala and Isabelle/jEdit. <http://www4.in.tum.de/~wenzelm/papers/async-isabelle-scala.pdf>, 2010.
- [16] Freek Wiedijk. Estimating the cost of a standard library for a mathematical proof checker. <http://www.cs.ru.nl/~freek/notes/mathstdlib2.pdf>, 2002.
- [17] Freek Wiedijk. The QED Manifesto revisited. *Studies in Logic, Grammar and Rhetoric*, 10(23):121–133, 2007.
- [18] Freek Wiedijk. Encoding the HOL Light logic in Coq. <http://www.cs.ru.nl/~freek/notes/holl2coq.pdf>, 2010.
- [19] Freek Wiedijk. A synthesis of the procedural and declarative proof styles of interactive theorem proving. <http://www.cs.ru.nl/~freek/miz3/miz3.pdf>, 2010.

Extended Abstract: Dynamic Proof Pages

Carst Tankink

James McKinna

Abstract

Reading a formal proof script written in a procedural (command-driven) style is difficult, bordering on impossible, without access to an interpreter for the script, that generates the states based on the commands. The Proviola system replaces the need for this interpreter, storing responses and displaying them on-demand.

This abstract describes a natural extension to the Proviola: instead of working on plain text files, the improved system takes an HTML rendering of the script, and decorates it with the proof states. The decorated page can be used as a Wiki page, generated out of a script file.

1 Introduction

Collaboration on formal (computer-verified) mathematics can be supported by taking a Wiki-based approach: a Wiki provides a repository of reusable results (*e.g.*, lemmas, theorems and definitions) and methodology, and it also lowers the threshold for contributing to the repository by novice users.

The software for such a Wiki can be designed in two directions:

- Bottom-up, or technology-centric, focussing on the technological problems one might encounter, such as consistency of the database, version control or file management.
- Top-down, or user-centric, first determining what activities to support for **authors** and **readers** of Wiki pages, by way of the interfaces (the pages in the Wiki) and developing the technology that underlies the interface.

Both approaches can be interleaved, as technological advances can inspire new interaction styles, while a fleshed-out interface drives the direction of technological development.

In this abstract, we will mainly work user-centric, describing a revised and redesigned approach to rendering the pages a reader might see in a Wiki for formal mathematics, henceforth simply a **MathWiki**. While we describe a design, the interface describes the behaviour desired of (formal) Wiki pages and can be considered top-down. Our work at present leaves open how an author can write such pages, both in terms of proof script content as well as Wiki commentary, but our approach emphasises upward compatibility, preserving and then enhancing existing author workflows.

1.1 Proviola

Because the MathWiki contains the artifacts of formal, computer-evaluated mathematics, it is possible to use the tools which manipulate these artifacts to enhance the Wiki pages. As one example of this, we developed the **Proviola** [6, 5].

The Proviola is a tool for displaying the dynamics of (script-based) formal proof development. A proof script is the result of an interactive session between an author and a proof assistant (PA). Such a session consists of the author writing commands to which the assistant responds with a proof state. The author proceeds by writing a new command, which modifies the proof state, and this game continues until the theorem is proven. The transcript of the session are the commands written by the author, which can be replayed by a reader at a later moment.

Because proof state is important in understanding such scripts, a reader typically cannot read the script without first loading it into the proof assistant and inspecting it step by step: the script represents

one side of the interaction (the author's), but to understand the proof coded in it, the reader needs access to the full interaction, including the PA responses.

When such scripts are published on the web, including as pages created for a Wiki, this may become a nuisance: the reader should have both a PA and a web browser open, using the browser to explore the links between notions, and using the PA to inspect the proof states and better understand the proof. To save the reader from having to flip between programs just to explore a repository, we create a new page for the Wiki. This page contains both the proof script and the state, as generated by the PA.

Declarative proofs An alternative to the Proviola would be to write proofs in a 'declarative' style, such as is done in the Mizar system [1] or when writing Isar scripts for the Isabelle PA [9]. Proofs in this style look more like traditional mathematics: the author provides the proof states and the rules from which these states follow, and the system checks that the use of these rules was allowed. This style can allow for readable proofs, but many proofs are already written in a procedural style, and it is our aim to make those proofs more accessible, without requiring authors to rewrite their scripts.

The tool that creates these dynamic pages is a **camera**: it parses a proof script, sends the commands to a PA and records the responses. The resulting list of (command, response) pairs is stored in an XML file, which is transformed into an HTML page with minimal JavaScript and CSS. This HTML page allows the reader to have direct access to the state, by just pointing at the commands he is interested in.

The first prototype for the Proviola worked with scripts for the Coq proof assistant, sending the commands through the ProofWeb [3] system. The resulting pages were undecorated, containing only the commands and responses as plain text.

Towards improving the display of the proof movies, we revised the tool to send each command instead to Coqdoc. This tool, part of the Coq [7] distribution, marks up commands and annotated comments in HTML, allowing syntax highlighting and hyperlinks to referred lemmas and definitions. In essence, for this version of our camera, we intended as output an HTML page as Coqdoc would render it,

but enhanced with the proof states as in our original movies.

This approach works, but has two drawbacks:

- It does not 'play nice' with other tools or workflows: the camera is a wrapper around Coqdoc, replacing that tool, so it cannot integrate easily into existing workflows that use Coqdoc produce HTML documentation, such as the "Software Foundations" course notes of Benjamin Pierce and collaborators [4]. Remedying this drawback is necessary to allow others to make good use of our tool.
- Because of the way Coqdoc works, namely on a per-file basis, instead of on a per-command basis, it is difficult to generate hyperlinks, which is essential to support readers in browsing a Wiki containing the enriched pages: the information Coqdoc uses to generate hyperlinks is stored in a separate file, that uses offsets in the original script for identifying location. This workflow presents each command as a stand-alone file to Coqdoc, uncoupling the presented data from the linking information.

We have recently re-evaluated our workflow for creating movies, to tackle both problems at the same time, and redesigned the tools accordingly.

2 Proviola: New Workflow

Instead of using the camera as a tool that calls Coqdoc, the camera was redesigned to be able to instead read the files generated by Coqdoc. Its implementation is straightforward: extending the parser for Coq

scripts to extract the commands from an HTML tree, by erasing the markup information. The resulting commands are sent to the prover, and the marked-up tree is stored together with the response.

Next to implementing a new parser, we also added communication with a locally installed Coq interpreter: instead of having to use ProofWeb, the camera can also use a local prover, allowing the use of special directives and libraries. We envisage that potential users and authors may wish to bind in custom PAs for document checking, beyond the vanilla distribution supported by for example ProofWeb.

This new workflow allows a camerawoman to create movies out of existing Coqdoc-generated HTML files, solving the first drawback. The second drawback is also solved: Coqdoc puts links into the HTML file, and these links are copied into the movie, possibly modified to refer to other movies.

There is a question of correctness arising from our redesigned process: how does a reader know that the movie is faithful to the original script? In the plain-text setting, this could be easily verified: the ‘command’ section of the movie should match up with the script. In the case of the new Coqdoc-reader, this requirement becomes a bit more complex: the reader should make sure that the transformation from script to HTML preserves the code, and that the transformation from the HTML tree back to the commands also remains faithful. To assist in this, the movie also contains the extracted commands, which might make it easier for the reader to verify this.

3 Movies as Wiki Pages

The Proviola camera can be used to generate pages in the MathWiki: it is a tool transforming a source file (the proof script) into an HTML page. This means we can generate a Wiki out of a repository of script files, where the proof states can be inspected on the pages themselves.

We have designed a prototypical Wiki system that creates HTML movies when a page is requested. It can be found at <http://mws.cs.ru.nl/mws/>. Because generating a movie can take some time, it runs as a background process, caching the generated movie for future use. Until that completes, the Coqdoc-generated HTML is shown. The movie-based Wiki page contains links to other Wiki pages (or pages in the Coq standard library). This prototype does not currently address any of the more technical issues, such as maintaining consistency of the repository when a page is changed, but serves as the basis for future research and experimentation.

The prototype has a basic editor: the entire proof script can be changed through a text box, where storing the script triggers a verification of the content, and clearing the cached copy of the movie. The content is only stored if it is deemed correct by the proof assistant.

This prototype does not provide any sophisticated support for (collaboration of) page authors, but focuses on the readers. These readers might become authors themselves at a certain point, but because formal development has a steep learning curve, they need all the help they can get in understanding the material that is already there.

4 Future Work

The Proviola is a reader-centric tool: it does not provide any new possibilities for writing the pages in the Wiki. To extend the Wiki, we intend to tackle the following issues:

- How to write a narrative of a movie, without having to change the underlying script; this use case arises when working with non-editable third-party sources. The sources might be non-editable because they are critical for other developments (similar to Wikipedia’s “protected pages”), but they might be included in an explanation of a formalization. Such narration might be achieved through some form Wiki-linking (specifying what objects to include in a marked-up document),

but to make this work for arbitrary regions of code raises some questions about how to specify the “comb”: what we previously [5] called the data structure(s) necessary to provide a commentary overlay which avoids having to modify the underlying source.

We do not intend this narration to be limited to describing a single proof script in a single file. Instead, a narrative can span an entire repository, or describe the different approaches taken in different proof assistants;

- How to relate such technology and workflows to those supported by other literate tools such as $\text{lhs2}\text{T}\text{E}\text{X}$ [2];
- How to extend this work to other provers: our prototype was easily adapted to the sequential interaction model of Isabelle, but this question still remains open for document-oriented PAs such as Mizar, Epigram or Agda, where the underlying interaction model is ‘batch-mode’ or even could be considered in terms of a ‘rectangle-based’ granularity of editing; in any case, such models challenge the simplicity of linear, script-based approaches;
- Integrate the prototype Wiki with the advanced methods for maintaining consistency in repositories. In particular, we will integrate the prototype with the version control-based workflows implemented for the Wiki for Mizar [8].

Our objective is to integrate such technology into a fully-fledged authoring framework, in which script authoring and checking, enhanced ‘Coqdoc’-style documentation, and third-party ‘narration’ co-exist within a MathWiki.

References

- [1] Mizar home page, 2011. Available through: <http://mizar.org>.
- [2] Ralf Hinze and Andres Löh. Guide to $\text{lhs2}\text{T}\text{E}\text{X}$, version 1.17. Obtained from <http://people.cs.uu.nl/andres/lhs2tex/Guide2-1.17.pdf> on May 30, 2011.
- [3] Cezary Kaliszyk. *Correctness and Availability. Building Computer Algebra on top of Proof Assistants and making Proof Assistants available over the Web*. PhD thesis, Radboud University Nijmegen, 2009.
- [4] Benjamin C. Pierce, Chris Casinghino, and Michael Greenberg. Software foundations. Course notes, online at <http://www.cis.upenn.edu/~bcpierce/sf/>, 2010.
- [5] Carst Tankink, Herman Geuvers, and James McKinna. Narrating formal proof (work in progress). In David Aspinall and Claudio Sacerdoti Coen, editors, *User Interfaces for Theorem Provers 2010. To appear in an issue of ENTCS*, 2010.
- [6] Carst Tankink, Herman Geuvers, James McKinna, and Freek Wiedijk. Proviola: a tool for proof re-animation. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D.F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *Intelligent Computer Mathematics*, volume LNAI 6167 of *Lecture Notes in Artificial Intelligence*, pages 440 – 454. Springer-Verlag Berlin Heidelberg, March 2010. Preprint available through <http://cs.ru.nl/~carst/files/proviola.pdf>.
- [7] The Coq Development Team. The Coq proof assistant. Web page, obtained from <http://coq.inria.fr> on October 5, 2009.
- [8] Josef Urban, Jesse Alama, Piotr Rudnicki, and Herman Geuvers. A Wiki for Mizar: Motivation, considerations, and initial prototype. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *AISC/MKM/Calculemus*, volume 6167 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 2010.
- [9] Makarius Wenzel. *Isabelle/Isar — a versatile environment for human-readable formal proof documents*. PhD thesis, Technische Universität München, Fakultät für Informatik, 2002.

Content-based encoding of mathematical and code libraries

Josef Urban
Institute for Computing and Information Sciences
Radboud University, Nijmegen

Abstract

This is a proposal for content-based canonical naming of mathematical objects aimed at semantic machine processing, and an initial investigation of how useful it can be, how similar it is to other approaches, what disadvantages and limitations it has, and how it could be extended.

1 Motivation

There are interesting naming problems when dealing with a large formally encoded mathematical library that is changing, in particular by parallel (nonlinear) editing. Some of the problems might appear also in large code libraries, either via correspondences like Curry-Howard and Prolog-like interpretations of mathematics, or just by the fact that code can have (implicit, explicit, chosen) mathematical semantics attached to it, like formal mathematics, and the structure of code libraries is similar to formal mathematical ones. Some of the problems (for example renaming of concepts) might appear also in not completely formally specified mathematical repositories/wikis, or this might also be relevant to semi-formal wikis where at least some part of articles has some semantic encoding. The problems are following:

Renaming: Mathematical objects might change their name, or have parallel names. Bolzano-Weierstrass theorem is often known just as Weierstrass theorem, Jaśkowski natural deduction [Jas34] as Fitch-style [Pel99], Jarník's algorithm [Jar30] as Prim's algorithm [Pri57]. Similarly for Solomonoff vs. Kolmogorov vs. Chaitin complexity vs. algorithmic entropy. Composition of two relations might be called `compose(R,S)`, or `rel_compose(R,S)`, or just `R*S`. Still, these names refer to the same mathematical content: The theorems have the same wording, and the functors have the same definition.

Moving: An item might be (as a result of wiki-like refactoring) moved to a different place in the library, which in many naming schemes can result in some kind of renaming. For example, a fully qualified name in the formal CoRN library [CFGW04] based on the Coq system could change from `CoRN.algebra.Basics.iterateN` to `CoRN.utilities.iterateN`. In the formal Mizar Mathematical library ¹, moving the fiftieth theorem in article `CARD_1` [Ban90b] to the first position in article `CARD_2` [Ban90a] would change its fully qualified name from `CARD_1:50` to `CARD_2:1`. Similarly for fully qualified names of functions in many programming languages. Any old term/formula/proof referring to the old name would later have to refer to the new name, making them look different.

Merging: Sometimes two or more objects might be merged into a single one. For example it might be recognized that Prim's description is essentially the same as Jarník's, and the two rewritten into a single unified form. Function composition might be recognized to be a special case of relation composition, and the two definitions merged into one.

¹<http://mizar.org/>

Note that whole hierarchies of such parallel notations and objects might be developed, possibly even within one sufficiently large library that has no tools for detecting such terminological parallelism. For example, it is quite conceivable that a large amount of parallelism exists when developing concepts related to functions and relations, like domain, range, inverse, transitive closure, etc. Similarly for multiplicative vs. additive groups/monoids, arithmetical operations/theorems on complex vs. real vs. rational vs. integer vs. natural numbers (when viewed as restrictions), leading to parallelly derived “different” versions of operations like power and modulo and parallel theorems about them using recursively different terminology.

How do we detect such parallelism and deal with it? Are we sentenced to re-invent pyramids of “more convenient” or “more fitting” or “more general” or “more politically correct” names of concepts over and over? And how do we find in the 1930’s vocabulary used in the developments by Jaśkowski and Jarník that they are isomorphic to other developments from 1950’s? Is there a matching/unification/subsumption/resolution/automated-reasoning algorithm that would (recursively) unfold the different concept names to some basic common language layer (set or type theory for example) without a significant performance penalty, allowing us to have easy search and inference tools for such heterogeneous libraries?

2 Content-based naming

It seems that the best the programming languages and formal libraries came up so far are module-based names, possibly enhanced by mangling the parameters and their types. The Mizar solution just follows the frequent mathematical practice of numbering definitions and theorems. Thus, the object naming typically depends either on human imagination, on the placement of the object at a particular place, or on both. None of these two guarantee stability with respect to the problems mentioned above.

The author is aware of three (related) solutions, each developed in a different context, listed here in the (assumed) historical order:

- Gödel numbering [Gö31]
- Recursive term sharing
- Recursive cryptographic hashing

2.1 Gödel numbering

The famous invention showing that arithmetic is self-referential. Every mathematical object is given a unique (impractically large) natural number as a name, based on an arithmetic function applied to its constituents (contents) already expressed as numbers. This obviously makes the name independent of placement in any particular module, and removes the human naming factor. As long as the wording (contents) is fixed, the name stays the same.

2.2 Recursive term sharing

In some computer implementations that deal with mathematical structures like terms (automated theorem provers (ATPs), Prolog) exhaustive sharing of terms is used to achieve space/time efficiency. In the E ATP system [Sch02], the terms $f(g(a))$, $g(g(a))$ can be printed using the following numeric representation (corresponding to how the system represents them as pointers): $a \rightarrow *0$, $g(*0) \rightarrow *1$, $f(*1) \rightarrow *2$, $g(*1) \rightarrow *3$. In this way, the number assigned to a term is unique², however,

²Obviously, something needs to be said about variables and their treatment, e.g as de Bruijn indices for this purpose.

it depends on the order in which a particular set of terms was presented to the system. If $g(g(a))$ was presented before $f(g(a))$, their numbers would be swapped. Because the numbering is contiguous, two such different numberings will be incompatible. So this naming is content-based like the previous one, however only inside one invocation of such a system.

2.3 Recursive cryptographic hashing

While the first scheme leads to impractically large numbers, the second is too fragile for practical purposes. Is there something providing the best of both worlds? An obvious direction in which this leads is minimal perfect hashing functions. However, constructing minimal (or small) perfect hashing function seems to be feasible only for finite sets of objects, possibly extended to uncomplicated denumerable sets of objects. There is no known (to the author) practical way how to have a reasonably well-behaving perfect hashing function for arbitrarily large complicated objects like mathematical formulas, terms, and proofs.

However, there is a practical approximation: cryptographic hash functions. Finding collisions for a cryptographic hash function like SHA-1³ or SHA256⁴ is so far very difficult, and “practically not happening”. This has been used for a number of purposes, one of the most recent ones bearing surprising similarity to the Gödel’s recursive encoding idea: SHA-1 based naming of files and directories in the Git version control system.⁵ To summarize, each file is by Git named as the SHA-1 hash of its contents, and each directory is named by computing the SHA-1 of the file containing the names, permissions, and SHA-1 hashes of its constituent files and subdirectories. This in practice means that Git only stores each duplicated file/directory once, and that some (both file and directory) renamings are very simple to recognize.

3 Content-based naming of formal mathematics

A particular modification of the Git’s SHA1-based internal naming scheme for Mizar (and similar formal/code libraries) could be as follows.

1. The initial library items (the set-theoretic equality and membership predicates in case of Mizar) are assigned an SHA1 value (e.g. their SHA1 value as strings, etc.) which serves as their unique name, that does not change between the library versions (all Mizar libraries are build up from the language of set theory).
2. A suitable semantic form is defined for terms, formulas, defined symbols, proofs, and other relevant library items. These all are trees (or DAGs), where the nodes are other items, and possibly some keywords of the language (if we follow Gödel’s numbering scheme, we might assign SHA1 values also to the keywords, punctuation, etc.).
3. The semantic form (tree, DAG of items - SHA1 values) is suitably combined to obtain the canonical value for the semantic form. In case of Git, this is for example done by sequential listing of files with their SHA1 value in a file, and computing the SHA1 value of the contents of such file. In our case, the most likely candidate for easy semantic representation is the existing XML format of all Mizar items (similarly for Coq) like definitions and theorems. We could either just compute a SHA1 value of each such item when treated as a file (with other items replaced with their SHA1

³<http://tools.ietf.org/html/rfc3174>

⁴<http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>

⁵http://book.git-scm.com/1_the_git_object_model.html

value), i.e. treating items as “files” in the Git internal naming algorithm, or we could do (probably more expensive) recursive computation of the value for all subtrees (similar to the shared term representation mentioned above). If such an operation turns out to be expensive, we could still use the fast local serial shared-term numbering to efficiently cache the previously computed SHA1 results for each instance of the library.

4 Proposed use, limitations, and extensions

A straightforward proposal is to compute such content-based names for all items in a large formal library like the MML or CoRN, and see how much naming-based duplication is inside the libraries. Sometimes such results could be surprising, showing that there are some significant redundancies, eg. when developing multiplicative and additive versions of algebraic structures.

Another direct use is for tracking the items’ histories during wiki-like refactoring: The library is in the case of the formal Mizar/CoRN wiki prototypes held in a Git repository already, and the file/directory-based SHA-1 hashes are very quickly computed each time a commit is made. Given this speed, it is conceivable that the above-explained item-based SHA-1 computation would not be significantly more expensive, and could be made an automated part of each repository commit (a particularly elegant way would be just to replace the Git default SHA-1 application with this one, and let Git use it internally instead). A side-effect of such an enhanced commit would be a mapping of human item names to the content-based ones, and a semantic diff report, saying which items have been moved and which have changed. This kind of experiment is probably readily feasible, for example on the recent one hundred versions of the Mizar library, or on the Coq-contribs repository.

Another immediate use would be for the search and automated reasoning/search tools over the libraries: A query to such tools would always be done in the content-based encoding. Thus, if a new user is (as is quite often the case with large formal/code libraries) ignorant of the particular naming conventions, and partially duplicates the concept hierarchy, this would not be a problem when asking some strong search/inference tools for an advice based on what is already in the library. The automated reasoning/search tools would work on recursive content-based encoding of both the library, and the new user’s queries and context, and provide the user also with advice about the relevant library names that should be used.

On the other hand, direct content-based naming is often unwanted. For example, in Wikipedia an article typically keeps its name for long time, even though its content changes. Such a stable name however naturally creates an equivalence class in the semantic space of SHA-1 hashes: some authority claims that a set of SHA-1 hashes have the same semantics according to some (stronger) point of view. Such equivalence classes could be used productively to allow human (or other) influence on the recursive content-based naming, propagating the equivalence classes using congruence-closure algorithms. This could serve as a semantic analogy of text tools like `diff`, which can be instructed to ignore white space changes. The user would in our case specify seeding equivalence pairs, whose (propagated) influence he would like the semantic diff (and strong semantic tools like ATPs) to ignore.

An interesting related problem is providing suitably normalized object representations, before they get consumed by the SHA-1 hash. For example, associative-commutative operators (like plus) are typically normalized into set-like representations (ignoring brackets and order) by ATP systems. This approach complements the previous one: semantic equivalence classes are not specified manually, but by specifying a content-normalization algorithm before the hashing function is applied. For semi-formal wikis, where for example only a small article/section part (say, the theorem) is semantically encoded, such normalizing function might consist just in ignoring the non-semantic parts of the articles/sections.

One limitation is that functions like SHA1 are only “practically” secure, not theoretically. If we are

very unlucky, we could for example infer a new theorem based on a clash between two differently defined concepts. An obvious remedy is to re-check the theorems in a safe encoding.

References

- [Ban90a] Grzegorz Bancerek. Cardinal arithmetics. *Formalized Mathematics*, 1(3):543–547, 1990.
- [Ban90b] Grzegorz Bancerek. Cardinal numbers. *Formalized Mathematics*, 1(2):377–382, 1990.
- [CFGW04] Luís Cruz-Filipe, Herman Geuvers, and Freek Wiedijk. C-corn, the constructive coq repository at nijmegen. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *MKM*, volume 3119 of *Lecture Notes in Computer Science*, pages 88–103. Springer, 2004.
- [G31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte Fur Mathematik*, 38-38:173–198, 1931.
- [Jar30] V. Jarník. O jistém problému minimalním (about a certain minimal problem). *Prace Moravské Přírodovědecké Společnosti*, 6:57–63, 1930.
- [Jas34] S. Jaskowski. On the rules of suppositions. *Studia Logica*, 1, 1934.
- [Pel99] F. J. Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20:1 – 31, 1999.
- [Pri57] R. C. Prim. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*, pages 1389–1401, November 1957.
- [Sch02] S. Schulz. E – a brainiac theorem prover. *Journal of AI Communications*, 15(2-3):111–126, 2002.

ProofWiki

Matt Westwood

1 Introduction

ProofWiki is an online repository for mathematical proofs. Its stated mission is: “the collection, collaboration and classification of mathematical proofs.”

To that end, it provides:

- A catalogue of definitions of mathematical concepts, categorised according to context (e.g. topology, number theory, analysis, graph theory, abstract algebra).
- A catalogue of axiomatic frameworks (e.g. Zermelo-Fraenkel axioms, axioms for Natural Deduction in the context of propositional logic, etc.)
- A repository of mathematical proofs, categorised, like the definitions, according to context.
- A rudimentary historical context, in the form of short biographical pages arranged into chronological order, in which the intention is to provide links to the original source works in which particular pieces of work first appeared.
- A glossary of works from which the material was sourced.

It can be found at www.proofwiki.org.

The screenshot shows the ProofWiki website homepage. At the top, there is a navigation bar with links for 'Prime mover', 'My talk', 'My preferences', 'My watchlist', 'My contributions', and 'Log out'. Below this is a search bar and a 'Go' button. The main content area features a large 'Welcome to ProofWiki!' message, followed by a 'Welcome to ProofWiki!' section with a description of the site's mission and a 'Quick Tips' section. A 'Latest Proof' section highlights 'Big Implies Saturated on 5 July 2011 by Qedetc'. A 'Top 10 Wanted Proofs' section lists various mathematical theorems and conjectures, such as the Gelfond-Schneider Theorem, Hausdorff Maximality Principle, Frobenius's Theorem, Fundamental Theorem of Galois Theory, Wedderburn's Theorem, Whitney Immersion Theorem, Whitney Embedding Theorem, Topological h-Cobordism Theorem, Thurston's Geometrization Conjecture, and Tartaglia's Formula. A 'News' section on the right provides updates from June 23, 2011, June 6, 2011, March 27, 2011, February 4, 2011, January 16, 2011, July 6, 2010, and March 14, 2010. The left sidebar contains navigation links for 'Main Page', 'Community portal', 'Current events', 'Recent changes', 'News Feeds', 'Random proof', 'Help', 'Donate', and various indexes and categories.

2 History

ProofWiki was conceived by Joe George, a student of mathematics at the University of Newfoundland, and set up in March 2008 by him with the help of a couple of friends.

Over the last three years it has been accumulating users steadily, until now there are over four hundred.

The author became involved in July 2008, having been alerted to it from mathhelpforum, to which he was a regular contributor. The author had a wealth of mathematical material which had been assembled in \LaTeX format over several years, but had been struggling to find a way to present it on the Internet. The still-embryonic ProofWiki appeared to be exactly the medium for presenting the accumulated material in the way it had originally been envisaged.

Since that time, the author has spent considerable time building up the site and contributing to the direction of its evolution.

2.1 Statistics

The following numbers may be of interest. They are accurate at the time of writing (5th July 2011):

- Number of proofs: 3795
- Number of definitions: 2819
- Number of users: 426
- From Google Analytics, between 4th June and 4th July:
 - Number of visits: 35,850
 - Number of pageviews: 88,866
 - Percentage of new visitors: 74.71%

2.2 Philosophy

A mathematician's work consists of proving things. How useful would it be, one would ask, to have a compendium of proofs of all sorts of standard results? No comprehensive website exists, we believe, dedicated entirely to the documentation of actual *proofs* of results. There are plenty of sites replete with definitions, but as far as we know, no site had yet been created with the express purpose of providing the proofs themselves.

Into how much detail should a proof go? Many published proofs contain phrases such as “It is easy to see that ...” or “It obviously follows that ...” and so on. It is the philosophy of ProofWiki to include all these skipped-over parts. Every step of a proof should be documented, and every concept defined. We consider that every page should ultimately be accessible to any reader who reaches such a page. The only barrier to comprehension is an understanding of the concepts. To that end, every single technical term is provided with a link to a page defining what it means.

It can be argued that the very existence of ProofWiki is a Bad Idea. In order to become a mathematician, one is supposed to be able to prove things. Much of the work of a mathematics undergraduate student consists of creating such proofs, from whole cloth, in order to develop the skills to progress. If such proofs are freely available on the Internet, and can be directly cut and pasted into place into an assignment document, then where is the benefit to the student?

This paper does not attempt to argue this point, beyond making the observation that all such encyclopedias cause exactly the same concern to be raised.

3 Structure

ProofWiki has been constructed with the following philosophical strategies in mind.

3.1 Separation of Definitions and Axioms from Proofs

Because a definition is a completely different concept to a proof, an early decision was made to separate the various categories of entity into separate namespaces.

These über-categories are available directly from the main menu sidebar, and can be browsed as follows:

- Proof Index
- Definition Index
- Symbol Index
- Axiom Index

... and so on.

There are other miscellaneous namespaces which are

3.2 Deep levels of categorisation

When you are crafting a proof in a particular field of study, you have a convenient folder where you can find any existing results relating to a particular definition.

Take for instance: Arens-Fort Space is Completely Hausdorff.

This result goes directly into two categories:

- Category:Arens-Fort Space
- Category:Completely Hausdorff Spaces.

In turn:

- “Category:Arens-Fort Space” is in Category:Examples of Topologies.
- “Category:Examples of Topologies” is in Category:Topology
- “Category:Topology” is in:
 - Category:Proofs
 - Category:Set Theory
 - Category:Analysis

- “Category:Set Theory” is in Category:Proofs

- “Category:Analysis” is in Category:Proofs

From the other path:

- “Category:Completely Hausdorff Spaces” is in Category:Hausdorff Spaces.
- “Category:Hausdorff Spaces” is in Category:Separation Axioms
- “Category:Separation Axioms” is in Category:Topology

... and so on.

3.3 Accurate and meaningful page names

There is a fine line between being too terse (thereby making it obscure) and being too wordy (thus making the category page more difficult to scan). We also try to use a “name” for a result, if it exists (and we can find out what it is).

As ProofWiki has evolved, so have the page names.

As an example, the following pages names have all been used for the Odd Number Theorem:

- Recurrence Formula for Square Numbers
- $1 + 3 + \dots + (2n - 1) = n^2$
- Sums of Sequence of Odd Numbers
- Sum of Sequence of Odd Numbers

Each of the superseded page titles have been kept as a redirect.

3.4 Short and pithy pages

Some wiki sites on the internet (in particular Wikipedia) try to make a page as comprehensive as possible: with a history, a list of examples, a paragraph of applications, a full dissertation of which directions research has taken the subject, and of course a summary for the lay person. ProofWiki’s philosophy is: less is more. If you have a great deal to say, then split it into several pages. One page: one result. One page: one lemma. One page: one specific example. (MediaWiki applications encourage the use of shorter pages anyway — if you write a page with over thirty thousand or so characters, it issues you a warning.)

Some topics, however, have many different aspects to them, to such an extent that to write these aspects all into the same page would cause that page to become larger than is desirable. ProofWiki has a way around this problem: using the inbuilt “transclusion” tool in MediaWiki, it is possible to construct a large page out of a number of smaller pages. This can be particularly useful when you want to group together many proofs or definitions in one place, but still keep the individual sections as standalone pages in their own right.

Some particular examples:

- Definition:Separation Axioms
- Trigonometric Identities

Note how each section title of these pages is a link which leads directly to the subpage from which the relevant parts have been transcluded into the main page.

3.5 A consistent approach to symbology

It is appreciated that there is considerable variation between various notations. While it is impossible to cater to everybody’s personal taste in notation, an attempt has been made to establish a notational convention on the site. When a concept is introduced by means of a definition, the various symbols for it are defined (if relevant with a historical perspective), and the ProofWiki preferred symbol is specified.

From Definition:Set Complement:

“No standard symbol for this concept has evolved. There are alternative symbols for $\mathcal{C}(S)$ and \bar{S} . $\mathcal{C}(S)$ is sometimes encountered, and may appear occasionally on this website. Another common one is S' , but it can be argued that the symbol $'$ is already overused.

“Some authors use S^c or S^{\complement} , but those can also be confused with notation used for the group theoretical conjugate. Some authors use CS . Another one is S^* , and another is \tilde{S} . You may encounter others.”

An attempt is usually made to use the notation which matches the name of the \LaTeX tag that defines it, for example $A \setminus B$ for $A \setminus B$ and $\circ \upharpoonright_A$ for $\circ \upharpoonright_A$. In cases where the notation used is not universally understood, the strategy is for every page which uses this notation to explain it wherever it is used.

It is planned (but no work has yet been done on this) for a list of “approved symbology” for the site. Till then we need to be tolerant of variation. The subject is contentious, as different users have their own preferred systems of notation. This is an area where advice may be needed.

3.6 The use of mathematical language over English

A specific example is that “iff” is preferred over the more wordy “if and only if”. The disadvantage of this is that casual viewers frequently see this as a typo, and well-meaningly “correct” it. This is alleviated by specifically linking to a page which defines what “iff” actually means.

3.7 A widely spaced presentation

A mistake often made in presentation of material for a computer screen is to mistake it for just another page in a book. However, it is believed by the author (and research may back this up) that in order to be properly taken in, information on a screen needs to be far less dense than on the printed page. To this end, a house standard is to present each thought (in most cases that means “each sentence”) on a separate line, with a blank line between it and the next line. More space can be put between rows to emulate paragraphs. Screen space is not paper to be conserved. Screen space is free.

An exemplar of this style is the page [Big Implies Saturated](#).

3.8 A historical perspective

The mathematicians who have contributed through the ages to this body of learning are credited as far as is known. Sources are documented (although this is currently incomplete) so that the user is able to access the original source material in which a concept is introduced. Variations in approach are documented when appropriate, and contentious and inaccurate statements are challenged.

Some exemplar pages:

- Definition:Pascal’s Triangle, in which a section of historical notes is included.
- Definition:Empty Set, in which an unwillingness by certain authors to acknowledge the existence of the empty set as an object is questioned.
- Union of Exteriors contains Exterior of Intersection, in which an inaccurate statement made in a published work is documented.

4 Infrastructure

ProofWiki was constructed around the readily available MediaWiki package (see <http://www.mediawiki.org>), a popular example of which is Wikipedia.

All mathematical material is written in \LaTeX according to a set of loosely-enforced house rules. This \LaTeX is rendered by MathJax, which is found on <http://www.mathjax.org/>.

ProofWiki has never been comfortable when viewed using Internet Explorer. It works, but the layout is compromised. For best results on a PC, Firefox or Google Chrome are recommended. The author has no knowledge of the user experience on other platforms.

5 Comparison with other sites

5.1 Wikipedia

See <http://www.wikipedia.org>.

As ProofWiki uses the same architecture as Wikipedia, and has been deployed with little customisation of appearance, the two sites have a similar look-and-feel. However, the strategies are markedly different.

Wikipedia, as its name suggests, takes an encyclopedic approach to its subject. A page on a given subject is designed to hold as much information as possible, with the result that pages can become so large as to become unwieldy and in extreme cases amorphous.

This is in direct contrast to the approach of ProofWiki, in which pages are deliberately crafted to be short and pithy. If there are multiple contexts for a given topic, then it is usual to split these up into separate pages, transcluded into a summary page. The MediaWiki software “transclusion” capabilities have been exploited to a fair extent to allow this to be accomplished.

As an example of how transclusion has been exploited, see the entry [Trigonometric Identities](#).

Also there is a high-level strategy to include on Wikipedia only “notable” subjects, and to disallow original research. On ProofWiki no such limitation has been imposed. If a mathematical item is interesting enough for a contributor to spend the time to enter it, then it is deemed worthy of inclusion.

Wikipedia is inconsistent in the technique for rendition of mathematical symbology. There is a continual discussion on that site as to whether it is better to use \LaTeX inline or whether to use conventional html formatting. It is the view of ProofWiki that *all* mathematics is best rendered in \LaTeX , in order to achieve full consistency of look-and-feel.

On occasion an article which has been slated for deletion from Wikipedia has been rescued and added to ProofWiki. A notable example of this is the entry on Boubaker polynomials, which had a stormy history on Wikipedia and was eventually deleted amidst a squall of wrangling.

5.2 Wolfram MathWorld

See <http://mathworld.wolfram.com>.

Wolfram MathWorld is arguably the most elegantly presented and comprehensive on-line mathematical encyclopedia, but then it is perhaps the oldest.¹ However, its focus is on definitions only, and no attempt has been made to add proofs.

In structure and presentation it closely resembles ProofWiki in its attempt to provide a compact and easily-assimilable page rather than bombard the reader with a large quantity of information. There is also a section on each page containing links to other relevant pages.

¹The author knows of no such older site.

It may well be suggested that the structure of ProofWiki was influenced significantly by MathWorld, but if this is the case then such influence was (at least in the mind of the author) completely unconscious.

Having said that, all such mathematics websites, not least ProofWiki, owe a considerable debt to MathWorld for showing what is possible.

5.3 PlanetMath

See <http://planetmath.org>.

ProofWiki bears a considerable resemblance in style, if not format, to PlanetMath. The latter has been evolving for considerably longer, so in many places is more comprehensive. However, the emphasis in PlanetMath is on definitions rather than proofs. While such proofs do exist, this is not consistent and therefore connecting the definitions with their uses is less than successful in places.

PlanetMath has been configured very much as a community, with an active discussion forum. On each page the author's username is prominent, as are the statistics for each user's contributions. Each page has a tool by which feedback may be offered on each page, but whether this is working or not needs to be reviewed.

Each page on PlanetMath also has its MSC number² appended to it. This is still a work-in-progress on ProofWiki.

5.4 The MacTutor History of Mathematics archive

See <http://www-history.mcs.st-andrews.ac.uk/history/index.html>.

This is a specialised site whose main purpose is to provide short biographies of all the major mathematicians in history. As such it has no intention of documenting any of the actual mathematics itself. However, it has been used as a resource for some of the historical detail which is often added to a page to add context.

5.5 Others

The author appreciates that there are many other websites out there whose content and approach overlap that of ProofWiki, but there has been insufficient research done in order to provide a worthwhile analysis of them. Their presence is welcomed and supported.

6 Who's Who

Because of its nature, nothing is known about the various authors beyond what they have seen fit to publicise.

There are three administrator accounts: "Prime.mover", "Joe" and "Alecscooper".

"Joe" has the responsibility for maintaining the website's infrastructure, for example, upgrading the MediaWiki software as it becomes necessary.

"Prime.mover" has contributed the bulk of the material on the site so far, and continues active and prolific involvement.

"Alecscooper" also contributes as and when his schedule allows, and keeps an eye on the user accounts.

²Mathematics Subject Classification

7 How To Contribute

ProofWiki has been set up to be completely free access. Anybody accessing the website has authority to add or amend any page at all. You do not even have to create a user account, although if you are preparing to contribute a significant quantity of materials you are firmly encouraged to do so.

This potentially leaves us open to all sorts of abuse to vandals and spam-merchants. However, this is guarded against by two so-far foolproof techniques:

1. A mathematical captcha: anonymous users are expected to solve a simple addition/subtraction test to prove they are not robots. While this looks as though it should be pitifully simple to subvert, so far it has worked.
2. The permanent disabling of perpetrators' accounts. This has been done in the past where users have set up accounts as a means to provide links to websites dedicated to non-mathematical activities (macramé, anti-smoking, political propaganda, and so on). It has also been done where users have performed acts of vandalism.³ Not only can the user account be blocked, but so can the IP address from where the dodgy account was set up.

The main reason why this technique has proved successful is because the level of traffic is low enough for one person to be able to monitor it completely.

8 Conclusion

The aim to provide proofs of every mathematical result ever published is clearly ambitious. The main limitation to our efforts is no more and no less than mathematical ability.

We need people who have the skills to be able to fill in the gaps. There are far too many “stub”, “work in progress” and “explain” pages that need to be completed.

8.1 ProofWiki constantly evolves

New results and interesting snippets are added as the contributors discover them. The readership grows wider, and ideas are continually being added for how to improve the presentation. Some may be good, some may not be so good — but the important thing is that the ideas are flowing.

8.2 There are no unbreakable rules

And that's another one. Most of rules and guidelines described in the structure section of this paper have been broken during the course of ProofWiki's evolution, sometimes because the page breaking that rule was written before the rules had evolved, and sometimes because in a particular context it “made sense at the time”. Sometimes a page is written by a person who does not quite grasp the philosophy upon which ProofWiki emerged (“it thinks, therefore it is”) — such pages stand as they are until they are rebuilt. Sooner or later.

³A case in point was where a user had a vendetta against another user, and made insulting and demeaning additions to various pages associated with that user. In that case the user, having been blocked, sent an email to the administrators begging to be allowed back on. The block was lifted, and the user continued to perform his childish acts of vandalism, for which his block was reinstated and set to last indefinitely.

8.3 ProofWiki is for everybody

All levels of mathematical literacy are catered for: from the straightforward simplicity of some of the proofs of Pythagoras' theorem to the forthcoming (planned — don't hold your breath!) full detail of Fermat's Last Theorem.

Join in!

WorkingWiki: a MediaWiki-based platform for collaborative research

Lee Worden
McMaster University
Hamilton, Ontario, Canada
University of California, Berkeley
Berkeley, California, United States
worden.lee@gmail.com

Abstract

WorkingWiki is a software extension for the popular MediaWiki platform that makes a wiki into a powerful environment for collaborating on publication-quality manuscripts and software projects. Developed in Jonathan Dushoff's theoretical biology lab at McMaster University and available as free software, it allows wiki users to work together on anything that can be done by using UNIX commands to transform textual "source code" into output. Researchers can use it to collaborate on programs written in R, python, C, or any other language, and there are special features to support easy work on \LaTeX documents. It develops the potential of the wiki medium to serve as a combination collaborative text editor, development environment, revision control system, and publishing platform. Its potential uses are open-ended — its processing is controlled by makefiles that are straightforward to customize — and its modular design is intended to allow parts of it to be adapted to other purposes.

Copyright ©2011 by Lee Worden (worden.lee@gmail.com). This work is licensed under the Creative Commons Attribution 3.0 license. The human readable license can be found here: <http://creativecommons.org/licenses/by/3.0>.

1 Introduction

The remarkable success of Wikipedia as a collaboratively constructed repository of human knowledge is strong testimony to the power of the wiki as a medium for online collaboration. Wikis — websites whose content can be edited by readers — have been adopted by great numbers of diverse groups around the world hoping to compile and present their shared knowledge.

While several somewhat high-profile academic wiki projects have been launched and later abandoned — the quantum physics community's Qwiki [14] for example — there have also been successful academic wiki projects that strongly suggest that wikis can be a transformative tool for accelerating and amplifying the power of research collaborations. In particular, the OpenWetWare wiki [4] has proven itself to have staying power as a home for an extended research community's projects and data, and the Polymath project [11] provides especially powerful inspiration regarding the power of online collaboration to accelerate the process of mathematical discovery.

MediaWiki, the software behind Wikipedia and its sister projects, available openly as free software, is especially powerful, full-featured, and stable, and is widely used in academic and popular sites alike. It excels at managing information organized into pages of text, as it is on Wikipedia. As such, it is very useful for collaborative and public documentation of a research team's techniques and results, but not applicable for collaboration on the daily research itself, which tends to involve writing of software tools for data analysis and simulation, and production of manuscripts for publication, with figures, tables, formulae and citations.

This paper describes a software package that extends MediaWiki, creating a hybrid environment which combines the desirable features of the wiki system — easy collaborative editing, recording of

history and authorship, and instant publication on the internet — with support for complex formats including programming languages and \LaTeX document formatting, making it possible to collaborate simply and flexibly on the actual daily work of the lab and making it simple to store and publish, in an integrated form, the results, process and presentation of the research.

2 WorkingWiki

WorkingWiki [27] is a software extension for the popular MediaWiki system that makes a wiki into a powerful environment for collaborating on publication-quality manuscripts and software projects. The WorkingWiki extension allows you to store “source files” in your wiki and develop, test, run and publish them easily, along with the products of computations using those source files. Examples include a project of five \LaTeX files and six EPS images that compile together into a single PDF file, or an R script that includes two other R source files and produces a CSV data file and several EPS figures. The WorkingWiki extension keeps track of when the source files have changed and when to redo the processing to update the output, and how to display the various file formats involved. The output files and images can be displayed in wiki pages along with the source code, and can be used as inputs to further computations.

Example: collaborating on a \LaTeX document. It’s very common for a group of scientific authors to write a paper by emailing each other copies of `.tex` files daily or hourly. This is inconvenient — in order to look at the paper you have to save the file into a directory and compile it — and unreliable — it’s easy to get mixed up and lose someone’s edits, or overwrite them with someone else’s copy of the file. One solution is to use a revision control system such as Subversion [8] or Git [2] to manage the source code, but if any authors are unwilling to take on the work of learning to use the tool, they’re likely to fall back on emailing the file or just dictating changes to someone else. WorkingWiki addresses this problem by providing basic revision control features together with easy editing. Once the `.tex` files, `.bib` files, and images are in the wiki, it’s easy for everyone to edit and see the updated results, and the wiki keeps track of all the changes and their authors, and makes it easy to review or undo them. It also provides a convenient place to discuss changes, without having to put comments into the manuscript itself, and can be used as a website to present the research to the public.

Example: collaborative, reproducible lab science. A research team can use WorkingWiki to archive experimental data (using the wiki’s history features to record who uploaded which data sets when); develop their data-processing scripts collaboratively in the wiki; construct the scripts that produce figures and tables in the wiki; create the manuscript that presents the results in the wiki; and finally export the manuscript as a `.tar.gz` file ready to submit to a scientific journal. The wiki can then be used to publish the data, source code, and manuscript to the world as is. This process captures all the files needed to understand and reproduce the research project, with its revision history intact, and in a form that is easy to annotate and publish online. A research team developing simulation programs rather than using experimental data can use WorkingWiki in the same way.

WorkingWiki is developed principally for research groups, but is likely to have a variety of other uses as well for mathematicians, scientists, and software developers. WorkingWiki provides some features of an integrated development environment: it coordinates compiling (if necessary) and running the code when relevant source files have changed, and displaying the results. It provides some features of a revision control system: it uses MediaWiki’s history features to record author, date/time, and content of every change to the files and the wiki pages they are connected to, and it allows viewers to export the source code to their workstations and work on it offline.

It integrates editing and running code with wiki editing. Source and product files can be mixed freely into wiki pages' text. Editing is fully collaborative. The effects of changes to source files can be fully previewed before saving to the wiki. The WorkingWiki-extended wiki is a simple, elegant way to present a research team's work to the world.

WorkingWiki has special features for translating \LaTeX documents to HTML, for display directly in the wiki page. WorkingWiki allows collaborators to edit complete \LaTeX documents collaboratively on the wiki, view the compiled document in the wiki page, and export the documents' source files to your workstation when ready to submit or circulate. Using Bruce Miller's LaTeXML software [20], the rendered contents of a \LaTeX document are made visible in the wiki page, including figures, citations and equations (optionally using MathML), as well as in the standard PDF format. The editing history of all files is maintained, including authorship of each change. WorkingWiki's \LaTeX handling works with documents that involve multiple files, stored on multiple wiki pages. \LaTeX `\include`, `\bibliography`, `\includegraphics`, and like commands are supported. Filenames do not need to match page names.

WorkingWiki is extensively customizable, supporting collaborative development and use of computer programs in any language. Images and other files created by computer programs can be included directly in \LaTeX documents and read by other programs, and are updated automatically when the programs or source data files are changed. The development environment can be customized by adding default make rules, and in many other ways.

WorkingWiki supports reproducible and open research by allowing researchers to collect all the files involved in a research project — data files, source code, documentation, publications — in an accessible place where collaborators can develop them together, and the public can be allowed to download the entire project, to verify results and try their own experiments.

3 WorkingWiki in use

WorkingWiki operates on *source files* that are stored in standard wiki pages. Source files are collected into *projects*. Behind the scenes, WorkingWiki maintains a cached working directory where it stores and processes the project's files. When an output file is called for in a wiki page or by other means, WorkingWiki does its work by invoking `make` [23] to create or update the file from the source files in its project before displaying it. In this way, users can edit their code (or their data files, or `.tex` documents) by editing the wiki, and run the code and view the output (the typeset version of the paper, the updated version of the figure, the textual output of the program) just by previewing or saving the page.

Figure 1 provides a simple example of the source text of a WorkingWiki-enabled wiki page. Most of the text of this page is standard MediaWiki markup using constructs such as `==...==` for section headers and `[[...]]` for links. A WorkingWiki source file is defined by including it in an XML-style `source-file` element. The text between the opening and closing tags of that element defines the content of that file, and it is written to the corresponding file in the project's working directory and used to update the project's output files as needed. In this example, project names are not explicitly given, signaling the software to use by default the project whose name is the name of the wiki page. The assignment of files to projects can be made explicit by supplying a `project` attribute along with the `filename` attribute in the opening tag.

Below the source file is an output file, represented by a self-closing `project-file` element. When the MediaWiki parser encounters this tag and passes it to WorkingWiki's code, WorkingWiki synchronizes all the project's source files with their copies in the working directory, creates a subprocess to run the Unix command `make figure.png`, and (assuming the `make` command succeeds) retrieves the file and inserts the file into the HTML page that is the output of the wiki's parser. Thus simply viewing the page causes the output file to be updated and displayed. Figure 2 shows what this wiki page looks like in

```
==R graphics example==
```

Here is a simple example of how to do a figure with R, using [[Recipe_Book#R | the lalashan site's custom rules for using R]]. The custom rules make it simple: just define a .R file:

```
<source-file filename=example.R>
plot(function(x){-x*cos(x-1)}, -pi, pi, col="blue");
</source-file>
```

and request its output using just the right filename:

```
<project-file filename=example.Rout.png/>
```

Figure 1: Source text for example WorkingWiki-enabled wiki page, illustrating the use of the `source-file` and `project-file` tags.

the web browser.

This example is especially simple because the steps to make the output from the source code are controlled by a system-wide makefile installed in a central location and used in all projects. This is not necessary: makefiles can also be added to individual projects, in the same way as any other source file, and in this way users can control the processing of any kind of files and specify their dependencies.

WorkingWiki completely supports MediaWiki's previewing feature: changes to pages, including source files, can be tested and revised extensively before saving them to the wiki. When a user previews a page that includes project files, WorkingWiki updates them from the modified source files, in a separate preview copy of the project's working directory. When the user saves the changes to the wiki, WorkingWiki merges the temporary files into the permanent copy, to avoid unnecessary repetition of processing steps.¹

Editing source code in a form field in a web browser is much less convenient than editing files in full-featured editors like `vi` and `emacs`, but browser addons It's All Text [15] for Firefox and TextAid [9] for Chrome make it much easier by allowing a page's contents to be opened in a text editor of the user's choice and kept open while repeatedly submitting and previewing the page or a section of the page. This gives users access to all the editor features they are used to, such as syntax highlighting and smart indenting.

3.1 \LaTeX features

WorkingWiki allows a source file to be displayed in a transformed form. For instance, if a page's editor writes `<source-file filename="example.R" display="example.Rout.png">`, the project file `example.Rout.png` is updated and displayed in the page in place of the source code. This feature has not proved very popular — it seems to be preferable to make source code visible in most cases — but the use of *default* display attributes is very useful with \LaTeX and related formats. Default display

¹When a directory becomes large, these copy operations can become quite expensive. Unfortunately, it's necessary, because if we processed unsaved code in the primary cache directory it could modify files in ways that would affect the outcome of future processing steps, even if the previewed changes were never saved. We partially address this problem by making it possible to split out large or numerous project files into separate project directories that are left uncopied provided the project's authors promise that they are protected from problematic side effects, but a more flexible solution would be desirable. We are considering using the Btrfs filesystem's copy-on-write file storage capabilities [1] to make these copy and merge operations fast and cheap.

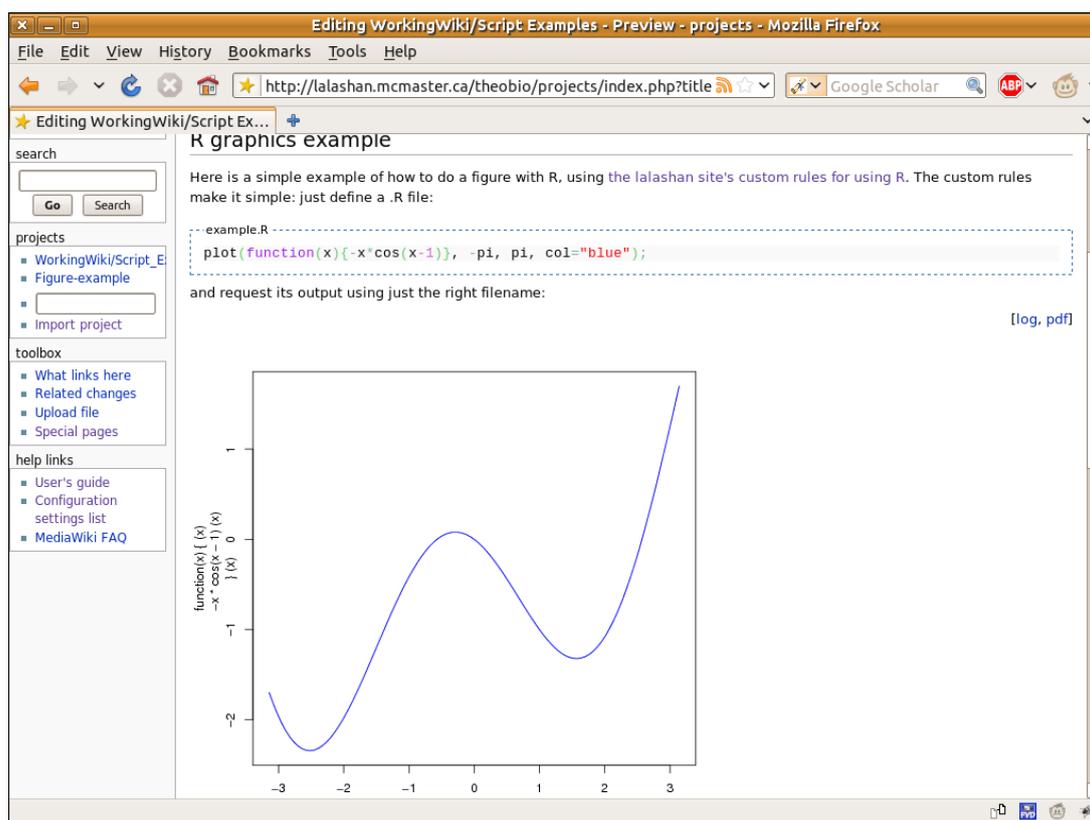


Figure 2: How the markup of figure 1 appears in a wiki page.

transformations can be defined by the wiki's administrators, and WorkingWiki comes with a small number of them predefined. In particular, `.tex` files are by default transformed to `.latexml.html` files for display, and WorkingWiki's system-wide makefile provides rules that make that transformation by using LaTeXML to process the document into HTML for display. Additionally, if a user has opted to enable MathML output and is using a MathML-compatible browser, WorkingWiki instead provides a `.latexml.xhtml` version of the document which uses MathML for all mathematical content. (This automatic detection and output switching is also available to wiki users and administrators for custom HTML-producing processes.)

When displaying a `.tex` file, WorkingWiki also provides a link that makes and provides a PDF version of the document using either \LaTeX or \PDFLaTeX (or other programs, if customized). This link can be redefined or additional links can be added to the default behaviour and they can be changed on a file-by-file basis by adding attributes to the `source-file` element.

In figure 3 is a screenshot of a \LaTeX document in a wiki page, illustrating how the XHTML version of the manuscript is embedded in the page, and the PDF link at the right margin. Embedding the rendered form of the manuscript directly in the wiki page allows for a comfortable cycle of editing, previewing, and editing some more, which is comparable to the ease of editing a manuscript's source text in a text editor, processing it into DVI or PDF, viewing, and editing again, especially when using an external text editor with the wiki as described above.

These features allow users to edit \LaTeX documents (and other source files) in the same way that wiki pages are edited: open an edit form and change the source code; press the preview button and see what it looks like when processed; edit some more until it is right, and then save.

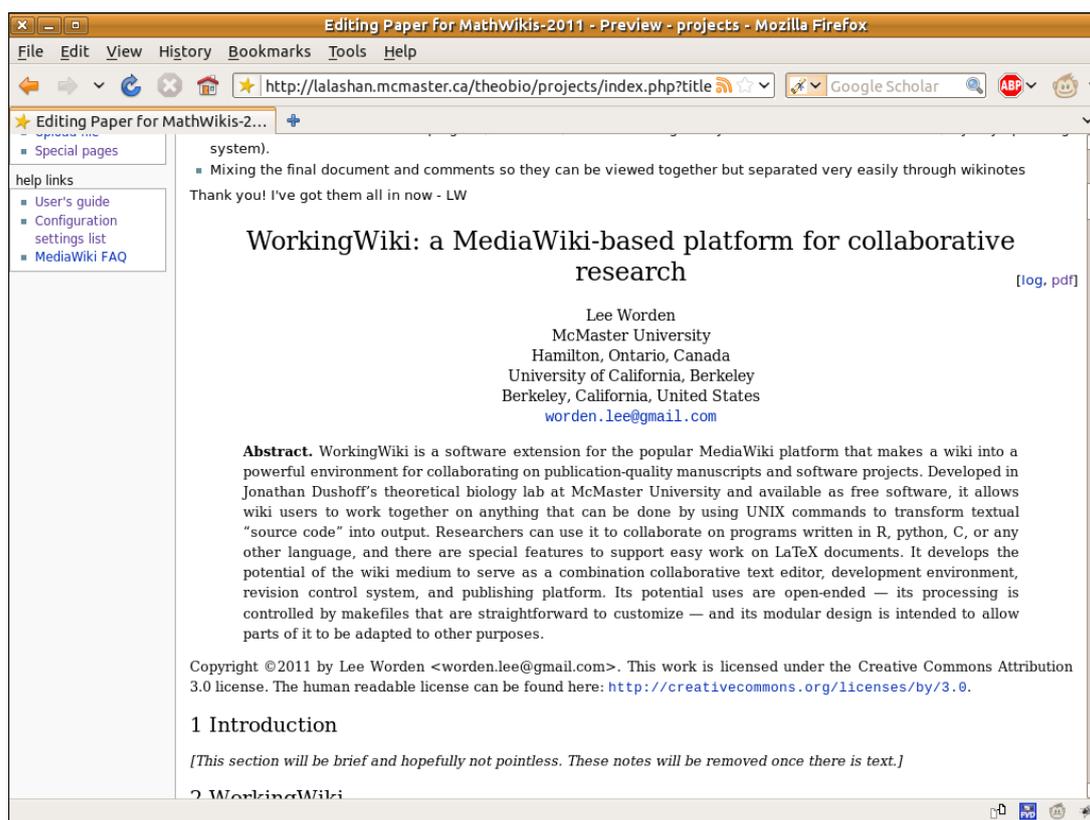


Figure 3: A \LaTeX document in a wiki page.

The make rules that are provided with WorkingWiki automatically keep track of dependencies on BibTeX files, figures, locally provided style files and included tex files, so that the displayed manuscript is kept up to date when any part of it is updated.

It is simple to use \LaTeX conditionals to insert comments and conversations in the code of a manuscript that are visible in the HTML version of the manuscript but invisible in the PDF, providing an easy way to coordinate while keeping a clean manuscript for submission.

3.2 Advanced features

3.2.1 Inter-project dependencies

For advanced users, WorkingWiki supports sharing of data among multiple projects, and takes steps to ensure dependency relationships are respected and data integrity is protected when previewing or running background jobs (see below).

This feature allows a number of useful strategies. General-use code can be shared among multiple projects, by placing it in a "library" project. Complex projects can be organized by grouping related things together into separate WorkingWiki projects, while allowing interaction between the different components. Independent parts of a project can be isolated from one another. A particularly important case is that a journal article for publication can be housed in a separate project from the data and programs that provide its content. This allows, on the one hand, the authors to maintain the dependency relationships within the wiki that allow the manuscript's figures and tables to be automatically kept up

to date when the data and programs change, and on the other hand makes it simple to export the article's source files in a neat `.tar.gz` package for submission to the journal and leave the programs and data behind.

3.2.2 Interaction with external data

WorkingWiki's back end is capable of processing data from multiple sources, and the front end allows those projects to be integrated with projects originating on the wiki. For instance, this author has a research project in progress in which a complex simulation program is stored in two GitHub repositories, pulled into two project directories in the wiki's file cache, compiled and run by code in a third project whose source files are housed on the wiki, and the output files are stored in a fourth project that is created on the wiki but doesn't have any source files.

This external-project feature also allows interaction between projects housed on different wikis — this is useful on our site at McMaster because we operate many interconnected wikis, and store some general-use code on a central wiki for use on others.

Project data can be exported to a user's local disk in a `.tar.gz` package, which includes the wiki's centralized makefile and other supplementary data, to allow running and developing the code offline. It can then be re-imported into the wiki. There is also a command-line tool to pull project files from wikis to a local directory. In the future there may be an interface to git [2], allowing one to pull and push source code from and to the wiki storage as if it were a (somewhat simplified) git repository. Given the flexibility of the git client, this would effectively make it possible to migrate projects easily between wiki storage and many other repositories.

3.2.3 Background jobs

When certain computational steps are too slow to run on the spot, WorkingWiki allows them to be run as background jobs, which run outside of the wikitext-parsing process. A background job is created simply by specifying a make target and requesting it be made in the background. Any background jobs that have been created are listed at the top of all pages that interact with the projects they involve, in a listing that provides their basic information and status. Whether a job has succeeded or failed, a user can browse its files, destroy it, or merge its output into the project's primary working directory. Running jobs can be browsed and killed.

In a standard installation, background jobs are run as Unix subprocesses on the same processors as the web server (using `nice` and `ionice` at the discretion of the site administrators), but there is a prototype in development to run background jobs on computing clusters using GridEngine [5].

4 Design of the software

WorkingWiki is implemented as a MediaWiki extension, written in PHP and augmented by a few JavaScript and CSS resources, makefiles, and small helper programs. It is freely available under the GNU General Public License [3], and is compatible with all versions of MediaWiki from 1.13 on.

The `source-file` and `project-file` tags are implemented as tag hooks, a standard means of extending MediaWiki's parser, and retrieval of binary files and project management are provided by two special pages, another standard form for extensions. Like MediaWiki, WorkingWiki itself provides a number of hooks that can be used by other extensions to provide additional features or modify the ones that are provided. It has not been tested in combination with all other MediaWiki extensions, but there are no known conflicts.

WorkingWiki's behavior can be extensively customized as is. Administrators can modify the rules controlling how different file types are displayed, and provide default transformations like the one from \LaTeX to HTML and links like the one from \LaTeX to PDF. Custom make rules can be added, to make it easy for users to write source code and transform it in standard ways, and the existing make rules can be partially or completely overridden.

4.1 Separation of wiki from project engine

A wiki is a powerful tool that combines a number of important functions. It is effectively a combined revision control system, integrated development environment, markup parser for website content, and publishing platform for web pages written in its markup language. WorkingWiki extends all of these functions to a wider range of source material, making the wiki into a combination revision control system, development environment, execution environment, and publishing platform for the general case of executable program text. Each of these functions is provided in more powerful forms by other tools, but the power of the wiki medium is in combining them together in an elegant, easy-to-use form.

An ideal situation would be to make it easy for end users to separate all these functions in a mix-and-match way, for instance providing a development, execution and publishing platform for data stored in a revision control system of the user's choice, or providing revision control, execution and publishing but using a third-party tool for editing and previewing. This is not entirely possible at present, but WorkingWiki is written with these separations in mind.

In particular, while the revision control, development (e.g. editing and previewing), and publishing functions are essentially provided by MediaWiki once the source files' contents are inserted into the stored pages and output files are inserted into the output HTML, WorkingWiki's execution environment is entirely separate from MediaWiki's code, and is designed as a completely independent component.

This component, called ProjectEngine, is a standalone tool that stores files, performs make operations, and serves up-to-date file contents. Written in PHP, as are MediaWiki and WorkingWiki, it can be used as a component of a larger program — it is incorporated in WorkingWiki in this way by default — and can also run as a self-contained HTTP service. It can be thought of as similar to a simple web server — whose primary function is to retrieve the contents of files for clients — but one that can create and update its files using make rules before serving them.

ProjectEngine supports updating and removing files; creating, destroying and merging preview sessions by making a copy of "persistent" files; and creating, destroying, merging and tracking background jobs.

The project engine seems to be a simple and powerful concept, and one that may have uses beyond this single wiki system. If nothing else, it can be used as a back end for similar extensions for other wiki engines, and the author has discussed this possibility with the author of Projects Wiki [19], a WorkingWiki-inspired plugin for Dokuwiki.

4.2 Security considerations

There are, of course, risks involved in running a web server that includes a project engine, which executes programs supplied by users. To a first approximation, the risks can be partitioned into just a few categories: overuse or destruction of server resources, access to sensitive data, denial of service, and harmful output. All of these risks can be managed.

The first category, overuse or destruction of system resources, is fairly broad. It includes scenarios from user-supplied code altering files on the server, to programs that send voluminous spam emails to innocent people, to infinite loops that consume excessive CPU time or fill up a disk partition. These risks can be managed by use of a mandatory access control system such as TOMOYO Linux [7] to restrict

access to all system resources, from sensitive files to use of the server's network interfaces. Additionally, ProjectEngine uses `nice` and `ionice` to prevent its processes from monopolizing CPU time and disk access, and uses `setrlimit()` to limit the number of subprocesses a make process can create and kills make processes after a limited time period. A quota system can be used to limit the amount of disk space ProjectEngine's files can consume.

Mandatory access control is also effective at keeping user-supplied code from reading sensitive system files, and WorkingWiki's inputs are carefully validated to prevent backdoor access to SQL data. WorkingWiki works with MediaWiki's access control features to ensure that a password-protected wiki doesn't reveal data to unauthorized users.

Denial of service attacks can include inputs that cause crashes in the software, as well as inputs that consume inordinate resources. The latter category has been covered. The former case can probably never be ruled out with complete confidence, but in any case, when a wiki is password-protected, unauthorized users have no means to interact with WorkingWiki and thus any attacks from outside the user community must be directed at other services.

The case of harmful output is the least well accounted for at present: in order to provide an HTML rendering of \LaTeX documents, it's necessary to allow ProjectEngine jobs to produce HTML output to be passed on to the client, and in order to support programmers it's necessary to allow them to write programs and custom make rules; in combination this means that users' projects can produce HTML output that does unwelcome things on the client side, such as making calls to third-party websites that reveal information about users logged in to the wiki. It may be possible to filter HTML output in a way that allows only safe output, but this is currently not implemented in WorkingWiki. Another possibility is to provide as an option a restricted set of WorkingWiki features, for instance allowing users to edit \LaTeX documents but not to create makefiles; this might suffice to provide a system that could be safely opened up to anonymous editors.

The current recommendation is to use WorkingWiki only on password-protected wikis, restricting editing access to trusted users. We believe it is safe for publicly readable wikis as long as only trusted users can edit. WorkingWiki is very useful and reliable for semi-closed wikis in this way, and use in public wikis more like Wikipedia may be possible in the future.

5 Examples of WorkingWiki in use

WorkingWiki's home site [27], which is itself a WorkingWiki-enabled wiki, provides a handful of example WorkingWiki projects, illustrating how to create projects for \LaTeX and for programming (the nomogram example [10] is especially engaging).

One active research team is using it to analyze and visualize African survey data related to HIV and female genital cutting. For that research a utility project has been created that automates the process of downloading the raw survey data from the provider's web site, merging separate data sets together, and transforming them into `.RData` files ready for processing in R. Another utility project provides custom R functions for plotting the data, allowing users to create visualizations of particular variables, including geographical plots, by inserting brief scripts of only a few lines into their wiki pages.

The Dushoff lab has created a suite of make rules to streamline R programming within WorkingWiki, making it easy to process data in steps by creating a series of small R modules that operate on the data produced by earlier modules, and interleaving these brief program snippets with the plots and textual output that they produce, in a wiki page that documents the steps of the processing. This mode of working with processing steps and their output is similar to the interactive notebooks provided by Mathematica [22] or Sage [6]. The Dushoff lab has also developed custom processes and make rules for automatically generating BibTeX data and browser-friendly reference lists from PubMed and similar

identifiers, making it easy to maintain citation data within the wikis.

This author is conducting an experiment in open research by maintaining a project on a publicly readable wiki. This project, which is in the early proof-of-concept phase, combines simulation and mathematical analysis in modeling collective search for a solution to a complex problem [24].

Another team is using it to investigate the behavior of spatially extended threshold models like those described by Schelling [21] and Granovetter [12], using a combination of python simulations and collaborative mathematical analysis (both in WorkingWiki). Other teams are using WorkingWiki to study the use of non-negative matrix factorization for community detection in marine ecometagenomics data, the effect of complex contact network structure on infectious disease dynamics, and spread of coexisting favorable mutations in spatially localized populations of plants and animals.

Papers completed in WorkingWiki have been published in *Ecological Economics* [25], *Theoretical Ecology* [13], and *Journal of Mathematical Biology* [16] (and now in this proceedings [26]).

6 WorkingWiki and math wikis

WorkingWiki's makefile rules are straightforward to extend or replace. Its \LaTeX features can be extended to additional formats: this author once created a structure for working on Sweave documents in WorkingWiki in an afternoon (it took a few minutes to write the rule to create `.tex` files automatically from Sweave files, and a few hours to create a LaTeXML style file to make the Sweave output look good in the browser). It should be straightforward to extend WorkingWiki to process any number of specialized document types conveniently; for instance, allowing users to edit sTeX [17] documents and render them automatically to PDF, browser-ready XHTML, and OMDoc [18] formats. It could also be used to develop, store, and process documents in computer-aided theorem-proving systems, or using any other tool that can be invoked from a UNIX command line to process files. Its uses are open-ended, and may prove very fruitful to explore.

References

- [1] Btrfs (b-tree file system). https://btrfs.wiki.kernel.org/index.php/Main_Page. Retrieved July 8, 2011.
- [2] Git: the fast version control system. <http://git-scm.com/>.
- [3] GNU general public license, version 2. <http://www.gnu.org/copyleft/gpl.html>.
- [4] OpenWetWare.org. <http://openwetware.org>.
- [5] Oracle grid engine. (Formerly known as Sun Grid Engine.) http://en.wikipedia.org/wiki/Oracle_Grid_Engine. Retrieved July 8, 2011.
- [6] Sage: open source mathematics software. <http://www.sagemath.org>. Retrieved July 8, 2011.
- [7] Tomoyo linux. <http://tomoyo.sourceforge.jp/>. Retrieved July 8, 2011.
- [8] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion*. O'Reilly, first edition, June 2004. <http://svnbook.red-bean.com/>.
- [9] Wayne Davison. Textaid. <https://chrome.google.com/webstore/detail/ppoadiuhggafnhokfkpphojggcdigllp>. Retrieved July 8, 2011.

- [10] Jonathan Dushoff. Nomogram. <http://lalashan.mcmaster.ca/theobio/math/index.php/Nomogram>. Retrieved July 8, 2011.
- [11] Timothy Gowers and Michael Nielsen. Massively collaborative mathematics. *Nature*, 461:879–881, 15 October 2009.
- [12] Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 1978.
- [13] Daihai He, Jonathan Dushoff, Troy Day, Junling Ma, and David J. D. Earn. Mechanistic modelling of the three waves of the 1918 influenza pandemic. *Theoretical Ecology*, 4(2):283–288, 2011.
- [14] Hideo Mabuchi lab, Stanford University. Qwiki (quantum wiki). <http://qwiki.stanford.edu>.
- [15] Christian Höltje. It’s all text! <https://addons.mozilla.org/en-US/firefox/addon/its-all-text/>. Retrieved July 8, 2011.
- [16] X. Jiang, J. S. Weitz, and J. Dushoff. A non-negative matrix factorization framework for identifying modular patterns in metagenomic profile data. *Journal of Mathematical Biology*, 2011. Epub ahead of print. <http://www.ncbi.nlm.nih.gov/pubmed/21630089>.
- [17] Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. sTeX – a system for flexible formalization of linked data. In Nicola Henze, Adrian Paschke, Andreas Blumauer, Richard Cyganiak, and Tassilo Pellegrini, editors, *Proceedings of I-Semantics 2010*. ACM, 2010.
- [18] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. Springer Verlag, August 2006.
- [19] Junling Ma. Projects wiki. <http://rsv.math.uvic.ca/dokuwiki/doku.php/development>. Retrieved July 8, 2011.
- [20] Bruce R. Miller. LaTeXXML: A LaTeX to XML converter, 2011. <http://dlmf.nist.gov/LaTeXXML/manual/index.xhtml>. Retrieved July 8, 2011.
- [21] T. C. Schelling. *Micromotives and Macrobehavior*. W. W. Norton and Company, 1978.
- [22] Wolfram Software. Mathematica: Technical computing software. <http://www.wolfram.com/mathematica/>. Retrieved July 8, 2011.
- [23] Richard M. Stallman, Roland McGrath, and Paul D. Smith. *GNU Make: A Program for Directing Recompilation*. Free Software Foundation, 0.71 edition, 2010. <http://www.gnu.org/software/make/manual/>.
- [24] L. Worden. Consensus dynamics project. Lee Worden Research Wiki, July 1 2011. http://lalashan.mcmaster.ca/theobio/worden/index.php/Consensus_Dynamics_Project. Retrieved July 8, 2011.
- [25] Lee Worden. Notes from the greenhouse world: A study in coevolution, planetary sustainability, and community structure. *Ecological Economics*, 69(4):762–769, 15 February 2010. http://lalashan.mcmaster.ca/theobio/projects/index.php/Greenhouse_paper.
- [26] Lee Worden. WorkingWiki: a MediaWiki-based platform for collaborative research. In *ITP 2011 Workshop on Mathematical Wikis*, 2011.
- [27] WorkingWiki home page, Jonathan Dushoff theoretical biology lab wiki. <http://lalashan.mcmaster.ca/theobio/projects/index.php/WorkingWiki>.