Joint Proceedings of the 24th Workshop on OpenMath and the 7th Workshop on Mathematical User Interfaces (MathUI) and CICM Work in Progress

Conference on Intelligent Computer Mathematics (CICM) Bremen, Germany, 9–13 July, 2012

edited by James Davenport, Johan Jeuring, Christoph Lange, Paul Libbrecht

14 October, 2012

1 Preface

This volume contains the papers presented at three subevents of the 2012 Conference on Intelligent Computer Mathematics (CICM; `http://www.cicm-conference.org/2012/`) in Bremen, Germany:

- the 24th Workshop on OpenMath
- the 7th Workshop on Mathematical User Interfaces (MathUI)
- the Work in Progress section of CICM

Both workshops were held on 11 July, 2012; Work in Progress talks were given on 9 July and 12 July.

The conference hosted further workshops, which publish their own proceedings; please see `http://www.cicm-conference.org/2012/cicm.php?event=workshops&menu=general` for an overview.

We would like to thank our peer reviewers for carefully reviewing the submissions and giving constructive feedback.

14 October, 2012
Birmingham

James Davenport
Johan Jeuring
Christoph Lange
Paul Libbrecht

Programme Committee

Chairs of the subevents are in bold.

MathUI Workshop

David Aspinall	School of Informatics, University of Edinburgh, Scotland
Paul Cairns	University of York, Great Britain
Olga Caprotti	University of Gothenburg, Chalmers, Sweden.
Paul Libbrecht	Center for Educational Research, Martin Luther University of Halle, Germany
Andrea Hoffkamp	TU Berlin, Germany
Elena Smirnova	Texas Instruments Inc. Education Technology, USA
Paul Topping	Design Science Inc., USA

OpenMath Workshop

James H. Davenport	University of Bath, UK
Lars Hellström	Umeå Universitet, Sweden
Peter Horn	metaminded UG, Aalen, Germany
Michael Kohlhase	Jacobs University Bremen, Germany
Jan Willem Knopper	Technische Universiteit Eindhoven, Netherlands
Christoph Lange	University of Birmingham, UK
Paul Libbrecht	Center for Educational Research, Martin Luther University of Halle, Germany

CICM Work in Progress

Johan Jeuring, Utrecht University and Open Universiteit the Netherlands, was the general programme chair of CICM. Work in Progress submissions were reviewed by the following people, representing all tracks of CICM (AISC, Calculemus, DML, MKM):

Jeremy Avigad	Carnegie Mellon University, USA
John Campbell	University College London, UK
Jacques Carette	McMaster University, Canada
Paul Libbrecht	Center for Educational Research, Martin Luther University of Halle, Germany
Claudio Sacerdoti Coen	University of Bologna, Italy
Volker Sorge	University of Birmingham, UK
Geoff Sutcliffe	University of Miami, USA
Josef Urban	Radboud University Nijmegen, The Netherlands
Matej Urbas	University of Cambridge, UK
Makarius Wenzel	Université Paris-Sud 11, France

Contents

MathUI

MathDox Select: A tool for creating SCORM packages from existing exercises

Hans Cuypers
hansc@win.tue.nl

Jan Willem Knopper
jknopper@win.tue.nl

June 24, 2012

Abstract

Learning Management Systems are often used in e-Learning and e-Learning content has to be made available for use inside such systems. This is usually quite easy to do. SCORM is a standard for this. However, if the content needs to run on a dedicated server, which might be different from the server running the LMS, this can be complicated. This is for example the case with interactive mathematical exercises in the MathDox system.

A tool has been developed to easily create basic SCORM packages that contain references to MathDox exercises. These packages contain scripts that communicate results on the exercises between the LMS and the underlying interactive exercise system (MathDox). In this way we have created a method for teachers to use MathDox exercises in SCORM compatible LMS like Moodle, ILIAS, or Blackboard without the need of a plugin.

As our set up is independent of MathDox, it should be fairly easy to extend this to other exercise systems or e-learning applications.

1 History

At the Eindhoven University of Technology MathDox has been developed [2,3,7]. This is both a document format and a set of tools to show interactive mathematical documents on the web. The last years MathDox has been extensively used to create and serve exercises to students [4, 6, 9, 13]. The exercises are offered through the Learning Management System (LMS) called Moodle [8].

We use a standard for serving e-learning content via an LMS: SCORM [11]. Some editors to create SCORM packages are eXe [5] and RELOAD [10]. However, the packages created by these tools are not suitable for e-learning content that, such as MathDox exercises has to run outside the LMS on a remote and dedicated server.

We have created our own tool, *MathDox Select* to easily create basic SCORM packages from a database of MathDox exercises, such that these exercises can

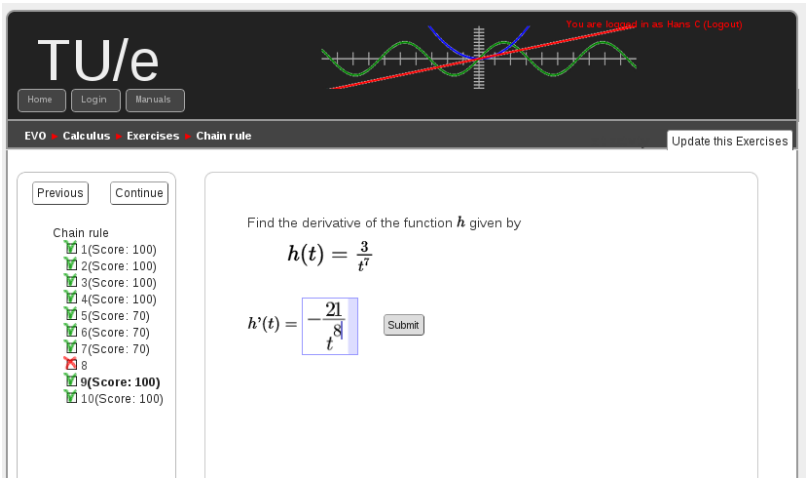


Figure 1: MathDox exercises running in a SCORM package within Moodle

run on our dedicated MathDox server, while the LMS might be running somewhere else. Our SCORM packages allow us to communicate user information and scores of exercises to the grade book of the LMS.

Since it is not necessary to install a plugin this means it is possible to add these packages as a teacher without administrative rights. The tool has been tested on Moodle and Blackboard.

As our setup is largely independent from MathDox, it can easily be extended to other types of e-learning content with scoring that one wants to deploy inside an LMS but has to run on a remote host. Here one can think of exercises from systems like ActiveMath [1], STACK [12] or WIMS [14].

With exercise systems it is important that the results of the exercises can be seen by the student and teacher. Exercises systems usually provide their own interface to results. WIMS and ActiveMath have their own interface, where you can log in. For teachers it is useful if they can see the results in their own LMS. Note that STACK has a plugin for Moodle, which is included by default in newer versions of Moodle. The advantage of our tool is that it only needs SCORM support to provides access to exercises and stores results.

MathDox Select was formerly called SPG (SCORM package generator) and earlier versions were written by M. Zubair Afzal and Matthijs Brouwer. This tool is being developed further for the ONBETWIST project [9]. Recent additions are support for multiple formats, metadata import, search functionality on metadata, and support to run exercises from a remote host.

2 MathDox Select

MathDox Select consists of a database of MathDox exercises (other types of e-learning content is also allowed), a web based front end to access the database, and a set of tools to create SCORM packages. Indeed, the front end makes it easy to select exercises and group them in a package. Such package can then be downloaded in SCORM format and imported into the LMS. The packages contain several scripts that make communication between the MathDox server on which the exercises reside and the LMS possible. In this way user information about the student available in the LMS is also available to the MathDox server, and information on the exercises, such as metadata and scores obtained by students, can be transferred to the LMS.

Inside MathDox Select the packages can be copied, shared with other users and grouped (to relate them to courses in the LMS).

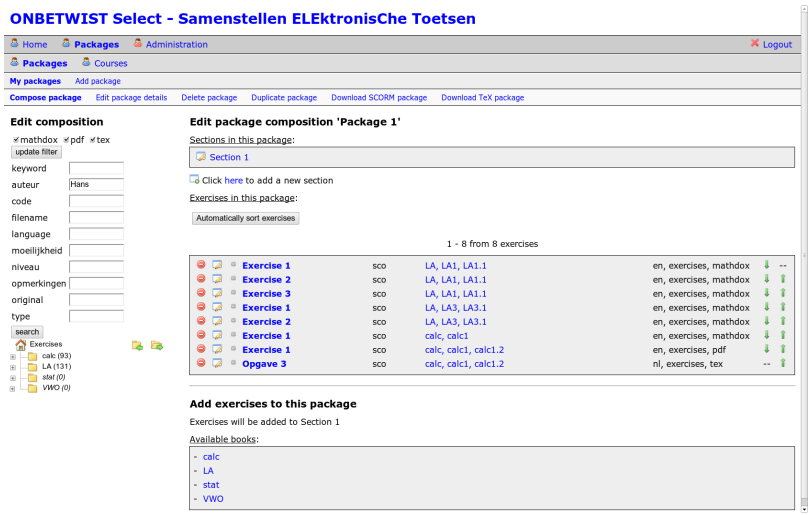


Figure 2: Selecting exercises and adding them to a package

3 Metadata and searching

The MathDox exercises in our database are XML-documents containing metadata based on the LOM-standard. Moreover, these exercises are ordered by a taxonomy.

The taxonomy is used to present the exercises inside the front end of the database in a tree structure. The database can also be searched by using the metadata.

To enable this search, our tool only shows the metadata fields from the

exercise sources that are actually used. As there are many metadata standards around, the choice has been made to allow metadata fields of any name and not just restrict to the metadata standard used within the MathDox exercises. In this way our front end is aware all possible metadata elements.

Because of this flexibility it is easy to use our tool for other collections e-learning content with existing metadata in possibly other format or standards.

4 The SCORM package

A SCORM package created by MathDox Select consists of HTML files with metadata together with some JavaScript files. The HTML files contain an iframe pointing to the exercises on the MathDox server. Since SCORM uses JavaScript for communication permission errors can occur when one wants to communicate between the LMS in which the packages are deployed and the remote server on which the exercises are running. The HTML5 solution is to use PostMessage to communicate between iframes. For this we use both JavaScript in the SCORM package and on a wrapper page on the server. It is easy to modify this JavaScript for use with exercises of another format.

5 Translation services

Besides the above described features, our tool is enhanced with some extra services. If exercises are the MathDox XML-format it is possible to automatically translate them into another format. An application we offer is conversion to LaTeX.

6 Multiple Formats

The set up of MathDox Select is almost completely independent of the MathDox system. As a consequence, it is possible to include e-learning content in various other formats in the database.

A similar approach as we have taken for MathDox exercises could easily be applied to e-learning content which consist for example of webMathematica pages, or of interactive exercises in the ActiveMath system or in WIMS.

References

[1] ActiveMath. `http://www.activemath.org`. Accessed: 30/05/2012.

[2] Arjeh Cohen, Hans Cuypers, Jan Willem Knopper, Mark Spanbroek, and Rikko Verrijzer. MathDox : A system for interactive Mathematics. In Joseph Luca and Edgar R. Weippl, editors, *Proceedings ED-MEDIA 2008 (20th Annual World Conference on Educational Media, Hypermedia and*

Telecommunications, Vienna, Austria, June 30-July 4, 2008), pages 5177–5182, Vienna, Austria, June 2008. AACE.

[3] Hans Cuypers, Jan Willem Knopper, and Hans Sterk. Mess: Exercises in MathDox. In *Electronic Proceedings of JEM meeting, Aachen, 2009*, 2009. Available from World Wide Web: `http://www.win.tue.nl/~hansc/mess.pdf`.

[4] Electronisch Verrijkt Onderwijs, TU/e. `http://evo01.win.tue.nl/moodle`. Accessed: 30/05/2012.

[5] eXe - the eLearning XHTML editor. `http://exelearning.org/wiki`. Accessed: 30/05/2012.

[6] Experience Mathness. `http://www.wistue.nl`. Accessed: 30/05/2012.

[7] MathDox. `http://www.mathdox.org`. Accessed: 30/05/2012.

[8] Moodle.org: open-source community-based tools for learning. `http://moodle.org`. Accessed: 30/05/2012.

[9] Onderwijs verBeteren met WISkunde Toetsen. `http://www.onbetwist.org`. Accessed: 30/05/2012.

[10] Reusable eLearning Object Authoring & Delivery. `http://www.reload.ac.uk/editor.html`. Accessed: 30/05/2012.

[11] Sharable Content Object Reference Model (SCORM). `http://www.adlnet.gov/capabilities/scorm`. Accessed: 30/05/2012.

[12] System for Teaching and Assessment using a Computer algebra Kernel. `http://www.stack.bham.ac.uk/`. Accessed: 30/05/2012.

[13] Technology Enhanced Learning of Mathematics for Master Education. `http://www.telmme.tue.nl`. Accessed: 30/05/2012.

[14] WWW Interactive Multipurpose Server. `http://wims.unice.fr/wims/`. Accessed: 30/05/2012.

Navigation in Mathematical Documents

Andrea Kohlhase

Computer Science, Jacobs University Bremen
http://kwarc.info

Abstract. Mathematical documents are not only hard but fun to write, they are equally hard but fun to read. Unfortunately, the "fun" part comes only in, when one has mastered the art of using mathematical vernacular and grasped all relevant context dimensions. Even though this seems an immanent issue for mathematical authors only, it is also for mathematical readers. In this paper, we argue for the need for more reader assistance in mathematical documents. In particular, we believe navigation to be a top candidate in this regard. Thus, we elaborate design opportunities for (semantically) helping the reader to navigate, which we anchor around the PlanetMath website.

1 Introduction

In recent years, the traditional notion of "document" changed drastically from a finished product accessible only to a selected group of people to a potentially open access, widely distributable, collaborative and flexible artifact. Documents are considered modern if they are **active documents**, i.e., documents that use a presentation engine to (re- or inter-)actively adapt the surface structure of the document to the environment or user input. Spreadsheets are the paradigmatic example for such an active document type: If a spreadsheet author adds a column to a spreadsheet, all cell references in underlying formulas are automatically updated by the presentation engine. Active documents exploit the distinction between form and content (like cell layout vs. computed cell values in spreadsheets).

Information on the Web is also often encoded in active documents (e.g. HTML- or XML-files). But Web documents frequently carry yet another property which non-Web documents don't have: The linear communication structures of traditional documents are replaced by multidimensional, multifunctional and multimedial hypertext structures, in which information is distributed according to the "Net Communication Model" (see e.g. [Flu96] or Sect. 2.1).

The central idea of this paper is the observation, that mathematical documents rely heavily on the Net Communication Model, which makes them hard to read without supporting services. Thus, we want to suggest potential reading services for modern math documents.[1] We first point out the underlying Net

[1] Please note that this is a workshop paper aiming at discussions around this topic. Thus, we do not consider the ideas fully worked out yet.

Communication Model in mathematical documents (Sect. 2). In order to read (and understand) a mathematical document, this either requires a highly active reader's mind or active documents that offer elaborate reader assistance. In Sect. 3 we look at an exemplary such service, which draws on the Net Communication Model in a specific category of active math documents: Semantic navigation in spreadsheets. Finally, we envision more navigational features in active mathematical documents like articles on the PlanetMath website[2] in Sect. 4.

2 The Net Communication Model in Math Documents

The idiosyncracies of mathematical vernacular have often been discussed. Many researchers started out hoping to find a nice model for it, so that the process of either reifying a human's mathematical knowledge into reviewable documents ("codification") or converting existing math documents into formal documents ("formalization") could be supported or even automated. The same kind of problem arises when math educators teach math: young mathematicians have to learn "*. . . becoming conversant in the language of mathematical discussion.*" [Day11].

Mathematical vernacular turns a math document into an **intellectual artifact**, i.e. according to [dS05, p. 10],

- *"it encodes a particular understanding or interpretation of a problem situation*
- *it also encodes a particular set of solutions for the perceived problem situation*
- *the encoding of both the problem situation and the corresponding solutions is fundamentally linguistic (i.e., based on a system of symbols [. . .]); and*
- *the artifact's ultimate purpose can only be completely achieved by its users if they can formulate it within the linguistic system in which the artifact is encoded"*

In particular, de Souza points out that intellectual artifacts require that producer and consumer use the same language. Math documents as intellecutal artifacts make use of a linguistic encoding system. As mathematical vernacular itself does not seem to be too systematic (otherwise all the research already done would have resulted in finding this system), it only seems to support the encoding system.

Therefore, in this section, we take a look at the underlying discourse structure of mathematical vernacular and find an underlying Net Communication Model.

2.1 The Net Communication Model (NCM)

Media-theorist Vilem Flusser suggested several discourse models for communication in [Flu96]. He distinguishes four communication discourse types:

Pyramid The **pyramid discourse** preserves information very well, as it is centered around getting confirmation about every obtained and possibly recoded information by its sender. Edited articles can serve as an example for this discourse type, as the information is checked by the author in the final proof copy before it is published.

[2] `www.planetmath.org`, relaunched shortly based on Planetary services (see [**?**])

Theatre The **theatre discourse** is characterized by a sender and a receiver exchanging information, protected against communication noise and thus misunderstanding by a concave theater screen. As long as the actors constrain to the rule to be within the protection area of the screen while communicating, the information entropy is very low. A document handed from the author to colleague in his field fits this description. The safeguarding screen here is the resp. Community of Practice (CoP) (see [LW91]).

Graph The tree or rather **graph discourse**, Flusser describes as sciences' discourse. The original sender (=author) creates information to be communicated, this information is broken down into information fragments, that in turn are picked up, re-coded and further communicated. Flusser calls it "centrifugal information distribution" and points to its inherent information creation while not conserving the information's original content. In many related work sections in scientific documents we can find such distributed information fragments, that give the setting for innovative ideas, i.e., new information.

Amphitheatre The final discourse type, Flusser presents is the **amphitheatre discourse**. Here, the information is neither send by one sender (but by many) nor is there a screen available against which the communication happens. This is the modern, web discourse type, the **Net Communication Model (NCM)**. The information entropy is extremely high, but information is communicated extremely fast and broadly.

2.2 The NCM in Math Documents

Now we want to look at the discourse type of math documents.

- **Pyramid:** As it is very important for mathematicians to conserve the "objectivity" and "truth" of statements, this safest-of-all discourse type seems highly attractive for mathematical communication in documents. Publication in a journal is still the most honorable publication format for mathematicians, and here the author indeed carefully checks its proof before it is published. But this is only a check of the form. The real question concerns a document's content, that is, whether used information fragments by other authors are checked by those authors? Here, the answer is no. In particular, the pyramid discourse type seems not to fit math documents.
- **Theatre:** The "objectivity" and "truth" in math documents are kept by a more or less rigid reviewing system. This is organized within the concerned Community of Practice, which serves as a theatre screen for the safety of the communication of mathematical information. Therefore, we conclude at first glance that mathematicians prefer to practice this communication discourse type. Unfortunately, this type does not support information creation, but mathematicians love the creativeness in math, for example:

  "*It is somewhat remarkable that a subject with such high and objective standards of logical correctness should at the same time provide such opportunity for the expression of playful genius. This is what attracts*

many people to the study of mathematics, particularly those of us who have made it our life's work." [Day11]

So the theatre discourse type doesn't seem to fit too well to math documents.

– **Graph:** But the graph discourse type supports creativeness, so we can consider this a fitting type for math documents? One of the most successful tools of mathematicians is their practice of aligning different discourses according to their purpose, for instance:

"*Mathematics is about* ideas. *In particular it is about the way that different ideas relate to each other.*" [Ste90, p. 2]

Moreover, modularization is another key mathematical practice, which enables fine-tuned aligning of theories. We conclude, that mathematicians use the graph discourse type in their documents.

– **Amphitheatre:** But how do mathematicians preserve "objectivity" and "truth" in their documents at the same time? They screen the information by their assumption of a math document to be self-contained. In particular, any reader can persuade herself of the truthfulness of inferences/conclusions. Therefore, even if a miscommunication happened in the discourse at some point, the derived statements are true in their local context. STEWART gave an example of the vernacularous depth of mathematics by letting a fictious mathematician (in a discussion with a fictional layman) exclaim

"*You can't get a feeling for what's going on without understanding the technical details. How can I talk about manifolds without mentioning that the theorems only work if the manifolds are finite-dimensional para-compact Hausdorff with empty boundary?*" [Enz99, p. 45]

This example demonstrates that the self-containedness has its natural limits. Those links outside math documents are part of the amphitheatre discourse type.

In summary, math documents are built on basis of the amphitheatre discourse type, but including parts of the theatre and the graph discourse types locally. In particular, they yield the Net Communication Model with local creativity-enabling provenances and local safe-guards.

3 Math Reader Assistance: Semantic Document Navigation

The difficulty of reading math documents becomes clear directly: To understand the locally used theatre screens, the reader has to be either part of the resp. Community of Practice or has to has access to its knowledge in all of its dimensions (see e.g. [KKL10]). Moreover, math documents frequently contain holes that refer to the implicit knowledge of the resp. CoP, which makes that specific part in the document potentially ambiguous based on the graph discourse type. This means that readers may fill in the missing parts incorrectly or not at all. Therefore, we argue for more reader assistance in mathematical documents in general.

3.1 Navigation

The showcased issues centered around a reader's orientation problems, thus, we consider navigation support a central service for these readers. In [IM00] navigation was introduced as a service that allows users "*to browse a huge document [...] without problems*". Originally, especially in the Nautics discipline, it describes the "*process of monitoring and controlling the movement of a craft or vehicle from one place to another*" [Wik11]. When transferred to the "*sea of information*" (e.g. [McI03]) available on the Web, navigation becomes the (elaborate) process of steering towards looked-for information goals.

Dourish and Chalmers introduced in 1994 the term "**semantic navigation**". In particular they stated that the

> *spatial organisation of data has been a highly visible component of a number of information systems.[...] Indeed, it's the notion of spatial arrangements which encourages (and legitimises) the notion of navigation of information systems. However, in navigation (as opposed to organisation), this use of the "spatial" is a convenient gloss for a different organisation, which we refer to here as "semantic".[DC94]*

They realized that in hypertext systems the primary form of navigation is of a semantic nature, in particular, depending on the semantic properties of existing links. Going beyond one-dimensional hyperlinks we therefore like to transfer this well-known notion of navigation to non-Web, active documents.

3.2 Semantic Document Navigation in SACHS

A possible way to provide reader assistance was presented for spreadsheets as active, mathematical documents in [KK11]. Even though they do not contain the usual mathematical vernacular, they also operate in the same discourse type fashion. Their local screens are built for the data of the document, their local provenance as well. Underlying formulae and charts are computed resp. drawn on the spot, so that the presented data are locally "true" data. The reader's interpretation of the data depend on the locally given, provenanced-dependent context information like headers. The known usability problems wrt. to spreadsheets were analyzed to be either false formulae, false data interpretation or input data errors (e.g. [KK10]).

We realized with the SACHS system [KK11] a semantic help system for spreadsheet documents, which includes interesting document navigation features, that we summarize next.

Let us start with introducing our running example document (see Fig. 1): An MS Excel spreadsheet based on [Win06], which can be considered a simple controlling system. It shows a profit-loss statistics over a time period, including cost and revenue data. The purpose and meaning of the spreadsheet seem clear enough, but as soon as we ask

– for which company this is a controlling system, or

	A	B	C	D	E	F
1	**Profit and Loss Statement**					
2						
3	(in Millions)	Actual			Projected	
4		1984	1985	1986	1987	1988
5						
6	**Revenues**	3.865	4.992	5.803	6.022	6.481
7						
8	**Expenses**					
9	Salaries	0.285	0.337	0.506	0.617	0.705
10	Utilities	0.178	0.303	0.384	0.419	0.551
11	Materials	1.004	1.782	2.046	2.273	2.119
12	Administration	0.281	0.288	0.315	0.368	0.415
13	Other	0.455	0.541	0.674	0.772	0.783
14						
15	Total Expenses	2.203	3.251	3.925	4.449	4.573
16						
17	**Profit (Loss)**	1.662	1.741	1.878	1.573	1.908
18						

Fig. 1. A Simple Controlling System Using MS Excel after [Win06]

– whether the numbers are given in millions of Dollars or Yen, or
– what the is definition of "projected data"

we realize right away that even in such a simple document the explicit knowledge is just the tip of the iceberg of the (background) knowledge necessary to interpret it. The SACHS system was designed to draw on a semi-formal formalization of this background knowledge as domain ontology.

The spreadsheet objects that carry meaning are the cells. They are interpreted by the user both wrt. the grid layout (like within a table with an assigned row and column specification) and via the underlying formula. With SACHS we offer a third dimension of interpretation by providing access to the background knowledge based on the alignment of cells with concepts in a domain ontology. In particular, we realized an embedded user assistance method by using cell clicks as entry points for the help system, that is, every click on a cell generates help.

Here, we are only interested in the help features "Dependency Graph" and "Search" box, since these two allow the user semantic document navigation.

The Dependency Graph

Let us first look at what happens when the user clicks on cell [B15] (Total Expenses, 1984). Then a new window (as seen in Fig. 2) is opened displaying at the top the concept connected to the selected cell. All concepts, which this top concept directly depends on, are shown on the second level.

If the user wants to elaborate on a specific concept like "Utility Costs", then a click on the corresponding node expands it by another level. The user is free to drill down into ever more abstract information available in the domain ontology; see Fig. 3 for an example path.

In a nutshell, the dependency graph enables the user to explore the background knowledge according to her own mental map of the concerned knowledge, her experience with it, her situation-dependent interests and time-frames. Cells are reinterpreted as hyperlinks to a domain ontology and moreover, the nodes in the dependency graphs are themselves links to further concepts in this ontology.

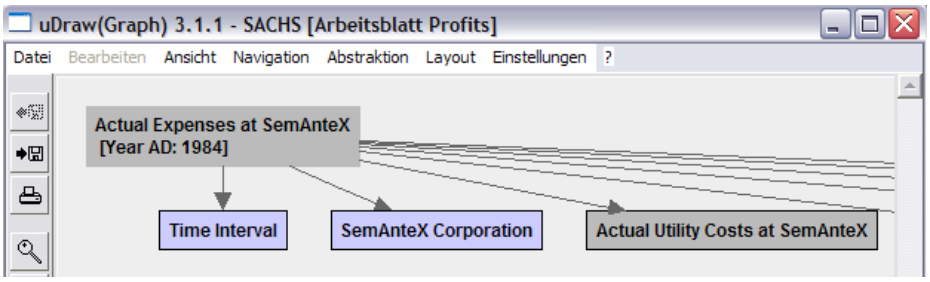


Fig. 2. Dependency Graph for Cell [B15]: Level 1 and 2

Iske [Isk02] distinguishes three navigational options based on a Network Communication Model: un- and directed browsing, and guided tours. A spreadsheet reader can explore the document and its previously hidden knowledge on her own, therefore SACHS enables *undirected browsing* as navigational service. Moreover, it offers *directed browsing* features as it is designed around the central semantic object for the interpretation of data within a spreadsheet — the cell (see [KK09]). The author of the spreadsheet and presumably of the resp. background knowledge governs the form of the dependency graph, thus the graph feature in SACHS can be also considered to promote paths or *guided tours* as a navigation option.

One can argue that this kind of navigation is mostly happening in the "document behind the original document": the background knowledge captured in a domain ontology. But the navigation functionality was also extended to cross-modality navigation by taking the alignment of concepts and cells as semantic document navigation cues.

In SACHS we mashed-up the graph-based interface with spreadsheet focus operations to enable **spreadsheet navigation** via the definitional structure of the intended functions in the structured background ontology. Note that the nodes e.g. in Fig. 3 have distinct colors. This color-coding indicates whether the concept in a node is connected to a specific cell in the workbook: An aligned concept node in the dependency graph carries a hyperlink to the resp. cell.

Let us look at Fig. 3 for example. There are two nodes in darker grey: "Actual Expenses at SemAnteX" and "Actual Utility Costs at SemAnteX". Clicking the node "Actual Utility Costs at SemAnteX" moves the spreadsheet focus to cell [B10]. Then the user can switch back to the original position in the spreadsheet by clicking the top node, here the "Actual Expenses at SemAnteX" node.

This way a user can get a good *orientation* on how the spreadsheet works and an overview over the various dependencies between cells.

Searching in the Background Knowledge

The search feature enables the user to reach through to the information available in background knowledge.

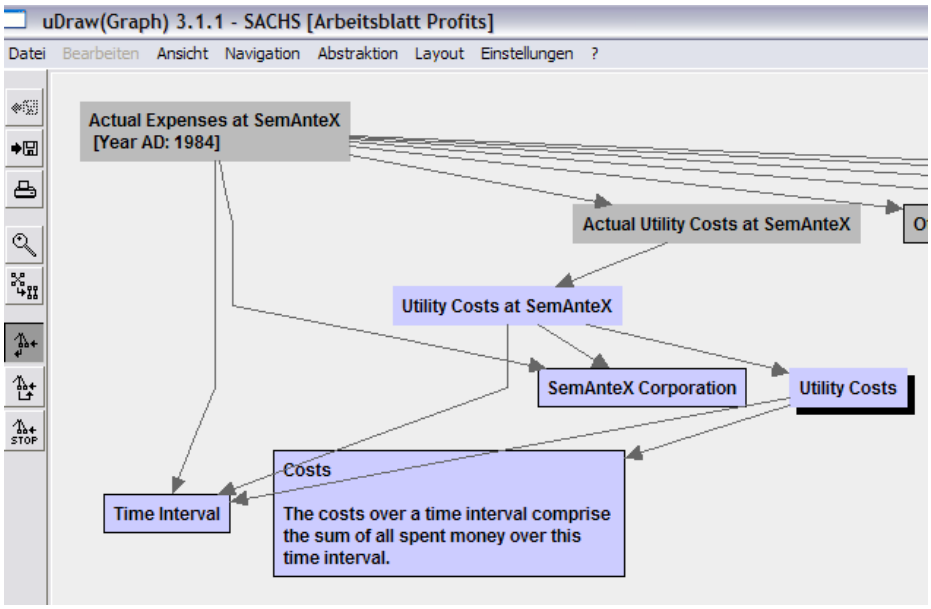


Fig. 3. Dependency Graph for Cell [B15]: More Levels

Concretely, users can type into the search box a string of characters which is used to search all concept titles for this string.

Once an element in the search result is selected by a user, the according explanation is presented in this area. For instance, she likes to know more about the concept "Steady State Prognosis", so she clicks on the according item in the list and its definition is shown (see Fig. 4). The concept "Actual Salary Costs per Time Interval at SemAnteX" is aligned to the cell [B9] in our running example. A click on the "Select Cell!" button would thus result in a navigation to this aligned cell.

On the lower right hand side of the "Search Results" area we can also find the "N!" command button. Pushing this results, on the one hand, in the selection of the next aligned concept in the search list, and on the other hand, in the navigation to the resp. cell in the according spreadsheet. The order is determined by the current cursor position within the list downwards and on user request starting on its top again. This feature enables the user who is unfamiliar with the concepts generally or concept titles particularly to judge by herself whether the shown concept is the one she was looking for to begin with.

In summary, we observe that both the concept graph as well as the hit list (the list of search results) can be used by the user as "navigation panels". The former is arguably more semantic (as it uses the conceptual dependency structure), whereas the latter is more mnemonic (it goes via the concept names). Their utility is largely due to the fact that they allow the user to focus on a particular

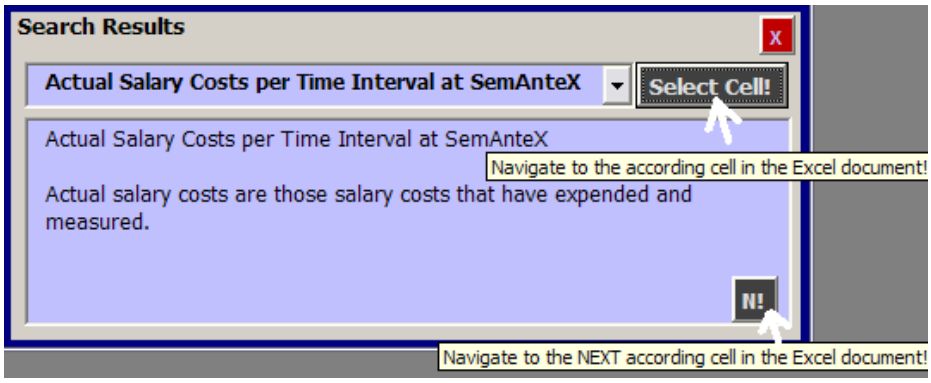


Fig. 4. Explanation for "Actual Salary Costs at SemAnteX" from Background Knowledge

aspect — the dependency cone of concepts leading up to the value of a particular cell or the concepts mentioning a particular string — and use these as a jump list for complex document collections. Indeed, commentators on the SACHS system always highlighted the added-value of being able to disregard worksheet and potentially even workbook limitations when navigating.

4 Math Reader Assistance of the Future: More Navigation

In order to envision more helpful navigation features, we imagine them for a web-based collection of documents as gathered on the PlanetMath website [**?**].

4.1 Articles Graph

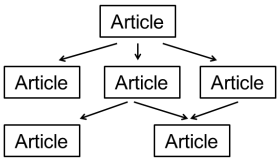


Fig. 5. Articles Graph

Consider for example the short article `http://planetmath.org/PrimeAn1.html`. The blue strings indicate links to other PlanetMath articles. The representation of the dependencies of this article on the other articles as dependency graph on the side, would give a clear indication of what this article is about. In contrast to the semantic document navigation in `SACHS`, the development of the graph level by level doesn't seem sensible as the user can do this by her own by using the hyperlink structure itself. But using this given hyperlink structure the reader quickly looses the overview and maybe doesn't even remember where she started at (think of the "magical number 7" in interaction design rules). A graph would be very helpful, in particular when deciding that a followed branch wasn't worth the effort.

4.2 Bibliographic Graph

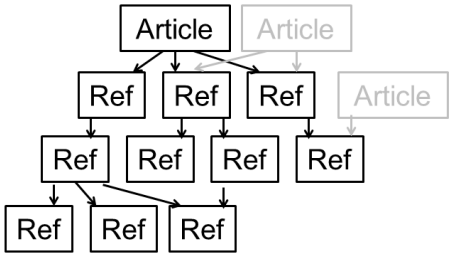


Fig. 6. Bibliographic Graph

Whenever one sees a list of bibliographic references, one is wondering which of all are the important ones. A hint could be delivered if the bibliographic references were presented in a "bibliographic graph", a graph containing all the references of an article and all recursively discovered references for each reference. The important ones are the likeliest to be referenced from the present references themselves (somewhat in analogy to Google's pagerank algorithm). If other articles referring to some of the present references, then this information may also be included into a graph. Naturally, this information could also be used to rank the references of an article.

4.3 Formula Graph

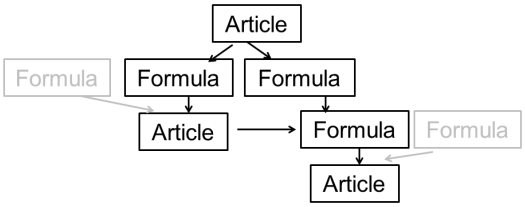


Fig. 7. Formula Graph

Other objects that appear frequently in math documents are formulae. If we take these serious as semantic objects, then a graph visualization also seems sensible. With the recent progress in math search facilities, structurally equivalent formulae can be found in other articles. Moreover, subformulae used in other articles can be discovered as well. If an article uses the same formula as the original article and additional ones, then these can also be used as a hyperlink in a respective graph.

5 Conclusion

In this paper we address the issue of the Net Communication Model together with some math specific extensions as a discourse model for math documents. This specific model makes it on the one hand hard to write and read math documents, but on the other hand it also provides the means to create a document that can only be understood from a holistic point of view. Once the author or the reader have been enabled to enjoy this view, fun is also part of the consuming process for that document. The holistic view depends on several dimensions of the underlying Net Communication Model. A math document can only be created if the author has mastered these dimensions, but not every reader has done so. Here, navigational features appear naturally as help services, as they

serve as access points to multi-dimensional information. We gave an example for semantic document navigation in spreadsheet documents via the SACHS system and envisioned some more navigational design opportunities for a collection of active, web-based math documents in the PlanetMath system.

We believe that such reader assistance will be particularly useful and gives them new, and efficient access to salient parts of math documents. In turn, this will induce a better overview and a deeper understanding of the concepts in users — and for some even enable an enjoyable reading experience of math documents.

References

Day11. Martin V. Day. *An Introduction to Proofs and the Mathematical Vernacular*. Online at http://www.math.vt.edu/people/day/ProofsBook/IPaMV.pdf, 2011.

DC94. Paul Dourish and Matthew Chalmers. Running out of space - models of information navigation. In *Proc. HCI*, Glasgow, UK, 1994.

dS05. Clarisse Sieckenius de Souza. *The Semiotic Engineering of Human-Computer Interaction*. The MIT Press, 2005.

Enz99. Hans Magnus Enzensberger. *Drawbridge Up*. A K Peters, Ltd., 1999.

Flu96. Vilem Flusser. *Kommunikologie*. Bollmann, Mannheim, 1996.

IM00. Naoko Ito and Kazuhisa Manabe. XML Document Navigation Language, 2000. http://www.w3.org/TR/xdnl.

Isk02. Stefan Iske. *Vernetztes Wissen*. Bertelsmann, Bielefeld, 2002.

KK09. Andrea Kohlhase and Michael Kohlhase. Semantic transparency in user assistance systems. In Brad Mehlenbacher, Aristidis Protopsaltis, Ashley Williams, and Shaun Slatterey, editors, *Proceedings of the 27th annual ACM international conference on Design of communication (SIGDOC)*, pages 89–96, New York, NY, USA, 2009. ACM Special Interest Group for Design of Communication, ACM Press.

KK10. Andrea Kohlhase and Michael Kohlhase. What we understand is what we get: Assessment in spreadsheets. In Simon Thorne, editor, *Symp. of the European Spreadsheet Risks Interest Group (EuSpRIG 2010)*, pages 111–121. European Spreadsheet Risk Interest Group, 2010.

KK11. Andrea Kohlhase and Michael Kohlhase. Spreadsheets with a semantic layer. *Electronic Communications of the EASST: Specification, Transformation, Navigation – Special Issue dedicated to Bernd Krieg-Brückner on the Occasion of his 60th Birthday*, 2011. accepted.

KKL10. Andrea Kohlhase, Michael Kohlhase, and Christoph Lange. Dimensions of formality: A case study for MKM in software engineering. In Serge Autexier, Jacques Calmet, David Delahaye, Patrick D. F. Ion, Laurence Rideau, Renaud Rioboo, and Alan P. Sexton, editors, *Intelligent Computer Mathematics*, number 6167 in LNAI, pages 355–369. Springer Verlag, 2010. http://arxiv.org/abs/1004.5071.

LW91. Jean Lave and Etienne Wenger. *Situated Learning: Legitimate Peripheral Participation (Learning in Doing: Social, Cognitive and Computational Perspectives S.)*. Cambridge University Press, 1991.

McI03. Neil McIntosh. World drowning in a rising sea of information. *The Guardian*, November 2003. http://www.guardian.co.uk/technology/2003/nov/01/internationalnews.onlinesupplement.

Ste90. Ian Stewart. *From Here to Infinity.* Oxford University Press, 1990.

Wik11. Wikipedia. Navigation — wikipedia, the free encyclopedia, 2011. [Online; accessed 20-September-2011].

Win06. Terry Winograd. The spreadsheet. In Terry Winograd, John Bennett, Laura de Young, and Bradley Hartfield, editors, *Bringing Design to Software*, pages 228–231. Addison-Wesley, 1996 (2006).

Skills Text Box A Tool to Access Resources by Mathematical Concepts

Paul Libbrecht
CERMAT, Karlsruhe University of Education

ABSTRACT

Searching for mathematical concepts being indicated in a document is not easy with the existing technologies because the current information retrieval technology has been designed for words and mathematical concepts are often made of more than single words.

Skills-text-box is an approach to this retrieval problem: it lets users use the classical search by words paradigm to search for the concept then identify it by choosing through a finite list. Skills-text-box is the device used to support the search engine of i2geo.net: at contribution and search time.

In this paper for MathUI 2012, we sketch the current technical development Skills-text-box, and present its advantages and limits as have been experimented by multiple mathematics teachers in Europe.

Introduction

Classical search engines are often insufficiently precise to search for mathematical documents. The approach presented in this paper attempts bridging the gap between a collection of learning resources which one knows well (such as a text-book in use for that year or an own collection of online resources), and a world wide web which tends to be a bit too wide (repeating itself often and rarely matching the exact expectations).

The way we tackle this enterprise is by a community platform, called http://i2geo.net, which proposes to share, annotate, and search for learning resources. This platform is meant to be easy to contribute and search and supports the fairly multilingual nature of geometric constructions by searching across the barriers of languages: e.g. a teacher is Spain should be able to find a resource contributed by a teacher in the Czech Republic without needed to speak that language.

This search and contribution approach supports practicing mathematics teachers which means that the annotations and queries are sufficiently fined grained so that *topics of next weeks course* can be expressed. For example, it is possible to differentiate resources training *right-angled triangle* and resources training *right-angles* and *triangles*: the precise nature of the mathematical concepts. We seem to observe that the mathematical science is very rich of these composite concepts and thus need a special treatment.

To this effect, the i2geo project, which finished in 2010, has encoded an ontology with the following concepts:

- a hierarchy of *topic*s relevant to the domain of mathematics
- a set of *competency*s object
- educational levels for each of the classes of Europe

Based on these annotations, the searches described above are feasible. This paper is a description of the central enabler of this search and annotation tool: a browser-component that supports the input of concepts of this ontology efficiently called Skills-Text-Box.

In this paper, we describe the detailed implementation and user-experience of this auto-completion component. It has grown between 2008 and 2010 in the Inter2Geo EU project. Papers about it include a description of the overall platform (I2Geo-DML-2009), of the overall project (inter2geo-CERME6). Its use has been successful at times but has also triggered reactions. We present the achievements and issues that have emerged after this experience, some of which were entirely unforeseeable during the planning phase.

Outline

We start with a description of the techniques used to achieve the objectives expressed above. This is followed by a detailed description of the user-experience in the current skills-text-box. Then we present achievements and limitations that we have met and open research questions that try to address these limitations.

Technical Underpinnings

The Skills-text-box library is a client and server software written in the Java programming language exploiting an ontology.

On the client, Skills-text-box is run in JavaScript: the usage of the Google Web Toolkit library (GWT) allows this compilation from Java into JavaScript code that works on the web-browser's object model. The client component displays a text-field. After about 200ms after the text-field has been modified, queries to the server are sent to suggest the nodes matching what has been typed. Once the suggestions have arrived, as an XML file, and are still valid, the result is displayed below the text-field offering the user to choose the node.

The data being searched is the GeoSkills ontology: a knowledge structure that represents topics as a hierarchy, competencies as a complex object (with verb and objects), and educational levels, pathways, and regions. GeoSkills is described in (GeoSkills-SemWeb-Handbook). Technically, GeoSkills is developed as an OWL ontology (an OWL DL ontology) which can be easily parsed and queried for. The ontology has grown along the years, reaching thousands of nodes which made it big enough to render unusable several editors. While Protégé was used, at the beginning, to edit the ontology, its usage has been

replaced a web-based tool that receives the contributions of curriculum experts in multiple languages: CompEd, a web-application that is running aside of Skills-text-box and is updating the ontology on a regular basis, see (CompEd-SWEL-2009) and the GeoSkills page.

The nodes of GeoSkills, including topics, competencies, and levels, all carry names so that they can be displayed to the user and can be queried for. The names are classified by frequency so that it is possible to rank their importance when searching through them: common-names are frequently used, less-common-name and rare-names are rather exception (but one should still find it if entering such a name), and false-friend-names represent names that should not be matched. Together, these names permit a quantitive ranking of the matches for a sequence of words as input in the text-box.

The server component of Skills-text-box is an index, based on Apache Lucene. The index stores all the names of each of the nodes. This indexing is performed by crawling through the ontology using the owlapi library and the pellet reasoner.

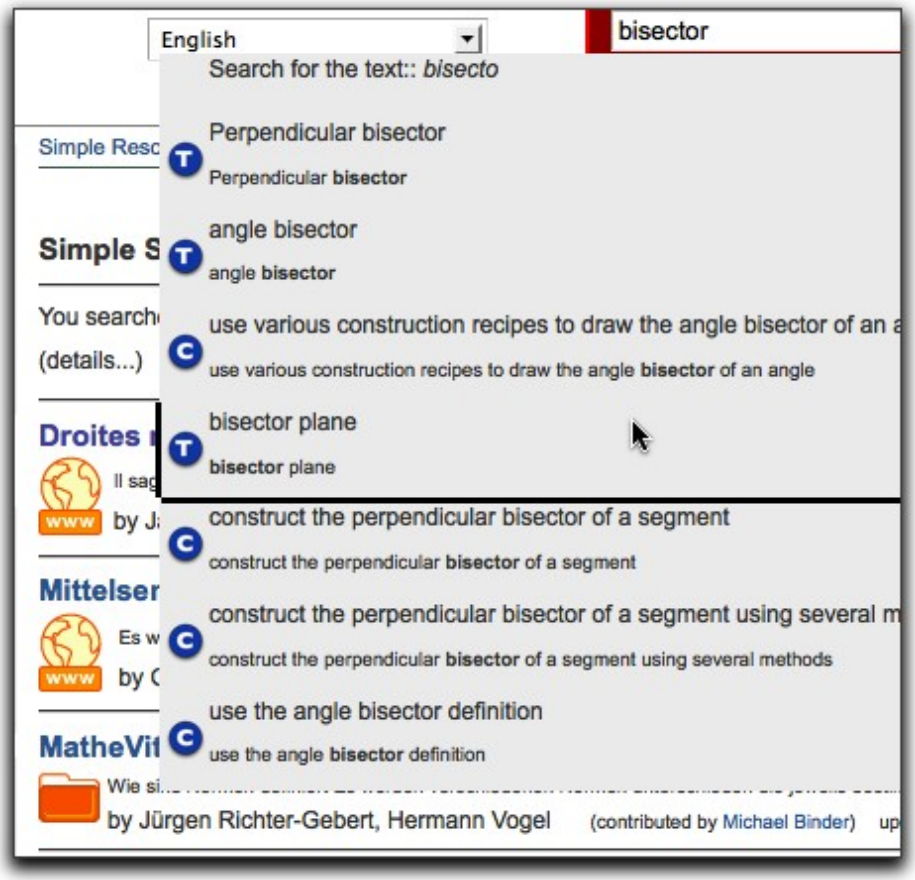
The server component is running as a separate web-application than the XWiki web-application which serves most of the i2geo platform's sharing needs (itself based on the collaborative asset management XCLAMS. Communication between the two web-applications allows the nodes to be rendered (by their name) and the auto-completion to trigger the search or contribution.

The SearchI2G server and client components, just as the CompEd API, and the i2geo customization to the XCLMS extension of XWiki are all available under the Apache Public License from the projects' pages. They are deployed and used on http://i2geo.net/ hosted by the Universy of Halle.

Current User Experience

When entering i2geo.net, users can search for resources by using the inputting a few characters in the search box on the top right. Soon after stopping to type, a waiting wheel indicates that auto-completions are being searched but do not prevent the user to keep typing: HTTP requests sent to the Skills-text-box auto-completion index are sent. When returned, after a duration of 1685 milliseconds in average, and the textfields hasn't been changed, the suggestions' popup is returned as on the left. The user can then choose the appropriate node of GeoSkills and trigger a search for the indicated node with either the mouse or the curosr and enter keys.

The auto-completions popup is made of the nodes of the GeoSkills ontology which, in order of preference:



- have a name equal to the the text searched for, end to end,
- have a name which contain words of the text searched for,
- have a name which contain words which start with the words of the text searched for

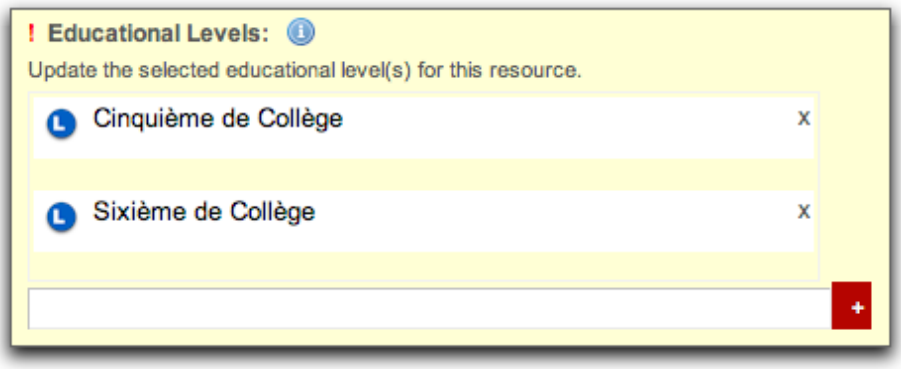
Each time, a name is searched for, one searches first the URI, the common-name, or the uncommon-name (providing a positive score contribution), or the false-friend-name (providing a negative-score-contribution). Moreover, one does this in each of the languages of the user: the language of i2geo being used (which can be changed on top of the page), the list of supported languages that the browser transmits through the Accept-Language header.

Each of the nodes are displayed by type: an icon indicates the type (level, topic, capacity), a small text indicating the name that was matched, and the *default common name* of the node. The default-common-name is a single name per node and language which allows a complete identification of the concept of the node even if the user is out of context: for example, *net of a solid* allows a complete identification enough while it is often searched for or named *net*. Similarly, the French naming of *4ème* (meaning *fourth class*) is insufficiently expressive, but *4ème du collège (France)* or *4ème primaire (Fribourg)* is precise enough; Skills-text-box allows the user to choose between both. Competencies are, typically, expressed as full sentences that contain a verb and its subjects in a way that

mimics the involved topics (for example: *calculate the slope of a line*) but queries search for them would typically not be the exact sentence but a few words approaching it, typically made of the "ingredients" of the competency (e.g. the concepts being manipulated).

If the user is unable to find the node, more typing is needed so that the results' list is made smaller: it seems to be impossible for an auto-completion pop-up to browse several pages of results. Moreover, only particular devices (those with a scroll-wheel or equivalent) allow the screen to be scrolled to view suggestions beyond the current screen. We shall see below that this challenge is a curation challenge that remains open.

Applications of Skills-Text-Box



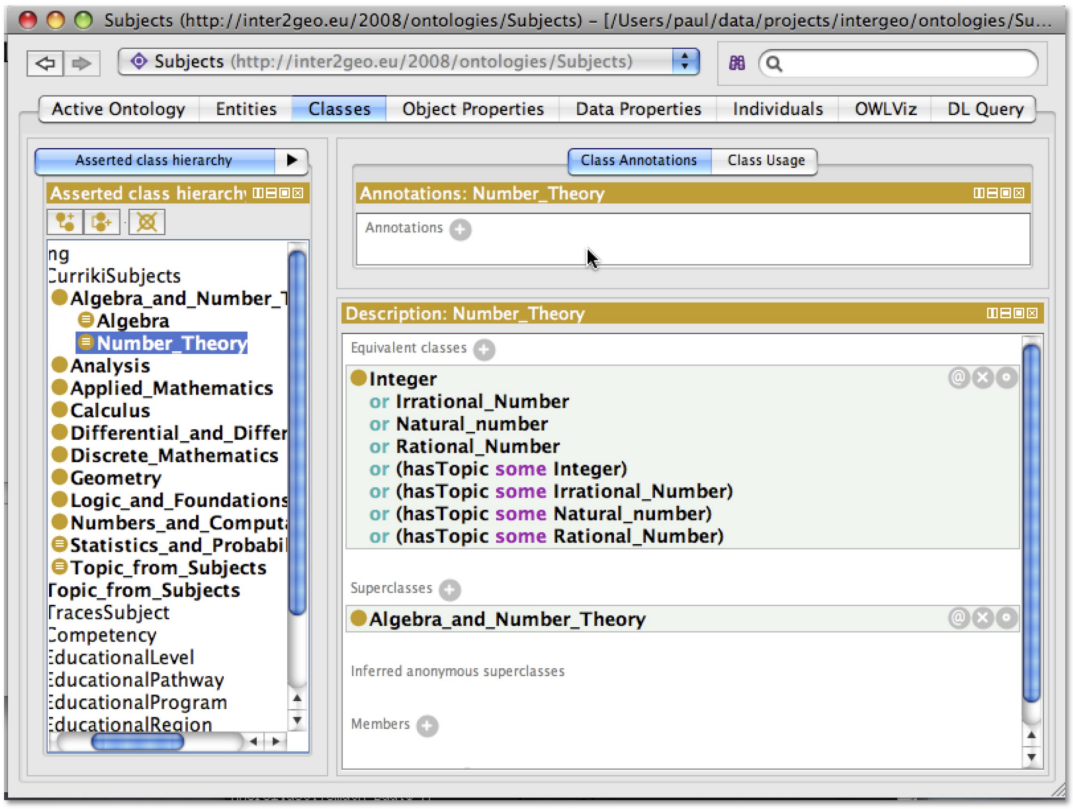
Skills-text-box is used to *speak GeoSkills*, that is to let the user express concepts encoded in GeoSkills: this is not an objective in itself, but it serves two objectives:

- let the user express a search query for a topic, competency, or level encoded in GeoSkills: this is done either as the sole clause in the simple search or as one of the clauses in the advanced search
- let the user annotate the resources when contributing them using the same language. In this case, as displayed in the screenshot on the right, the user's choices add subjects, competencies among the "trained topics and competencies" or levels among the "target educational levels".

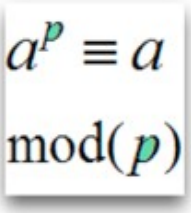
The search exploits the ontological nature of the GeoSkills queries further: if queried for a topic node of the ontology, it also queries for any topic node that is an instance of the class of this topic. This way an *equilateral triangle* is found when a *regular polygon* is searched, however resources annotated with *regular polygon*, in their generality, are preferred. Searching for competencies does a similar generalization: it searches for resources which are also annotated for the same competency-verb and the same objects.

Finally, the ontological nature is used in the subjects' search: subjects are encoded using the ontology editor Protégé by the usage of axioms which allow a fine description of the collections of topics and competencies. The screenshot below shows the editing of the

subjects as axioms using the Protégé 4 editor and the [choice of subjects in the i2geo platform]:

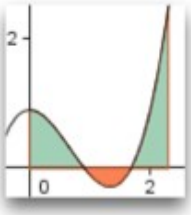


Browse Resources by Subject

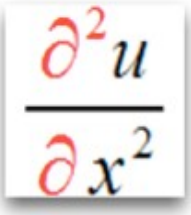


Algebra and Number Theory (7)

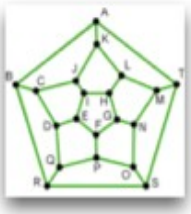
Applied Mathematics (24)



Calculus (57)

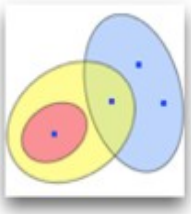


Differential and Difference Equations (0)



Discrete Mathematics (0)

Geometry (739)



Logics and Foundations (5)

Numbers and Computation (121)

Statistics and Probability (17)

Analysis (0)

Successes and Challenges

The approach to auto-completion and the challenges of cross-language and cross-curriculum search have been sketched at the very start of the Inter2geo project, together with all stakeholders. Its implementation, including the development of the ontology and its knowledge input by curriculum experts in each country, has been realized during the project.

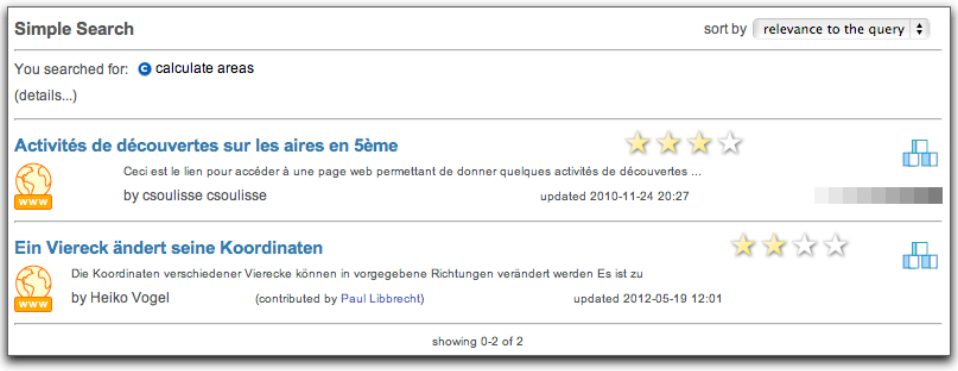
The search and contribution tool has been in regular uses by multiple teachers, in France, Spain, Germany, and the Czech Republic during the Inter2Geo project. This gave rise to feedback, sometimes close to rejection and sometimes enthusiastic. For some, this feedback was described as a log-book describing the multiple attempts at searching and the following evaluation of the applicability of a learning resource for subsequent teaching. This feedback has lead to incremental enhancements the final state of which is described above. Among the crucial feedbacks that came is that plain text search is still an essential feature that should not be discouraged.

Between February and June 2012, after the i2geo project finished and its usage was mostly spontaneous, 3398 auto-completions requests were responded while 2417 simple searches and 303 advanced searches were performed.

Successes

The skills-text-box function has succeeded at least under the perspective of its original missions to provide means to express annotations and queries for elaborate multi-words concepts: indeed, one can search for the phrase ~~angle droit~~ yielding 2 results, search for the words ~~angle droit~~ yielding 498 results (among which a fair amount which are not about right angle), search for the concept ~~angle droit~~ yield 1 result that is precisely an exercise about this elementary mathematical topic.

A query by concept is the way to formulate a query and find results in multiple languages. For example, querying for the text *calculate areas* and selecting the suggested competency, gives rise to two matching resources, both of which are in a different language than English. A screenshot of the result is below:



Limitations

Current users, however, also noted imperfections in the approach. Skills-text-box was often indicated to be insufficiently easy to use for the following reasons:

The set of terms that were most difficult to work with are educational levels: in countries where many educational levels exist (e.g. one per state is Germany). This certainly is due to the fact the states have not yet been enriched with common names, an issue which is related to the fact that the inter2geo consortium, which ran the project between 2007 and 2010, have long expected government agencies to provide us the list of educational levels. However, even if provided such a list of names, it is not clear that it will be easy to select levels because of the large overlap in naming between a *siebte Klasse* of the Gymnasium of Baden-Württemberg and Hamburg,

A topic is useful if it helps to find a category of learning resources which otherwise would not be possible. What to do with a GeoSkills' node that gives no matches? Currently, many nodes of GeoSkills match no learning resources in i2geo. They have been contributed to GeoSkills following an analysis of the learning standards, using the same words that these texts use. However, no-one has contributed a resource about it. A potential strategy is to hide such terms from the search (but not from the contributions' forms) but it has not yet been attempted because of the cross-application nature of such a implementation.

Different displays of the ontology are probably also needed and have been partially implemented:

- for example the tree of topics or the tree of competencies (present in CompEd but too slow to be useful)
- annotated displays of the curriculum standard for a few countries, either in the faithful reproduction of the official text, or as a javascript tree. They allow the teachers using them to find resources annotated with topics and competencies related to parts of the standard. we could not conclude on the utility of this approach.

Ongoing Research

Having described the technical foundations, the achieved user experiences and their limitations, we describe here open investigations which we intend to tackle during the Open Discovery Space project that just started and will federate multiple repositories of learning content through Europe, including i2geo.net.

Formal User Testing

So as to raise the utility of the search engine, it should be possible to apply classical testing methodologies of information retrieval such as those presented in Manning-Prabakhar-Schütze, chapter 8: through a formal approach, it is possible to obtain quantitative measures of the utility of the search engine and reproduce this approach having addressed issues reported in a qualitative fashion. The application of a formal testing should be done related to the "utility" of the search engine for the day-to-day of teachers who often measure the quality of a learning resource by criteria unexpected to computer-scientists (see the quality approach in I2Geo-Quality as one of the evaluation methodologies).

This practice would support, for example, refining the exploitation of the ontology structure into the search by guaranteeing that an added tolerance does not introduce too much visible noise.

How to better use the context?

There is a strong potential to make better use of the context of a user of the i2geo platform.

Indeed, it would be normal for a registered user to indicate his country of origin, and thus avoid to make it precise that the *4ème* (fourth class) is that of France. Such a refinement would be an extra step in the query expansion and should probably not exclude other types of *4ème*.

Beyond elementary query additions, one could "make closer" terms that are in curriculum standards of interest to the user. This can be done because of the property `belongsToCurriculum` which can be computed from the curriulum standards, a set of html pages that link to search queries for the given nodes.

Such an approach could be the right approach to respond to a request that we have never been able to implement: respect the different wording of the educational standard. Examples include the wording of the same competencies in different countries such as Luxembourg and France: for many, the competencies are equivalent, however the wording is different.

Strategies for the Maintenance of a Classification

An area where we have found surprisingly little support from the broad semantic web or digital library research infrastructure is on the long-term maintenance of the ontology: while several methodologies for the development of (new) ontologies can be found, little literature is available about maintaining an ontology: we have learned the hard way such rules as to avoid to change identifiers as soon as the ontology may been referenced elsewhere: for example, URIs are kept readable so that they run the risk of containing typos which one considers natural to fix for example. We would expect infrastructure and best practice markup to indicate that a node is kept there only for the sake of completeness but should not be referenced in new annotations. Such deprecations could, for example, be "sufficiently documented" so that user-interfaces such as i2geo.net would suggest replacement nodes the next time the user edits the resource.

Curation of The Search Results

Should the maintainers of the platform run regular gardening activities on the search results? One of the strategy could simply be to go through each of the nodes of GeoSkills, search for it and compare this search result with a search result for plain-text. A strong difference there, provided a clash of the meanings is not occurring (such as *angle droit* and *angle* and *droit*), could mean one of the following issues:

- that there has been insufficient annotations contributed
- that contributions have been done to other topics but this one was not considered necessary
- that the search engine is not doing the right generalization

This curation process is a community process as it involves raising the quality of the results for the use of everyone. Would sharing the search results-pages, since they can be exchanged by a URL, leverage this aspect and let people talk about it and invite them for action?

Curators could, following tests above:

- request a change to the programmes (e.g. change the way generalizations are made or displayed)
- request a change of the ontology (for example add the common names of educational levels or suggest to differentiate too unprecise concepts)
- mail the contributors of resources considered insufficiently tagged and propose an annotation enrichment
- publish the search as one of the test cases for future testing
- share a good search result as part of a demonstration action

Among such features, the i2geo project attempted to accept suggestions: on places where the user can contribute, a little "+" sign allows the users to formulate a suggestion so that

the editors of the ontology, which have the overview, can incorporate or invite to use another term. 34 suggestions were formulated during the project, with better results achieved when using direct communication to the curriculum-encoders, the community of workers that edit GeoSkills.

Finally, it is certainly the role of a community curator to listen to requests for content and evaluate if it is relevant and applicable to the platform and, if yes, start the appropriate contributions showing best practice that others can follow: indeed, quite often external contributors come to i2geo.net and expect to find particular topics but they do not leave a visible trace of such quests, especially if unanswered. Sharing such a quests, in the form of search URLs and accompanying texts in social networks or emails is a way to raise awareness both about the platform and the platform potential. This should be stimulated.

It is interesting to note that most of these actions are triggered only because of a particular state of the available data (the ontology and the annotated resources) and have been probably not identified as requirement in the development phases: the search paradigm is, indeed, strongly influenced by its available data; it is easy to make corpora which are impossible to search with ease because the words one would use as queries are not discriminating enough.

Fuzziness Approaches

Probably one of the hardest curation situation is when two GeoSkills nodes are meaning something similar to each other, but not exactly the same. Different communities will tag different resources with different nodes leading to the isolation of communities.

One way to exploit fuzziness is to let the user walk around: using graphical displays of the relationships between the GeoSkills nodes can support the user into generalizing or specializing his or her request. While this is currently available today, going through CompEd and navigating the tree, it is not smoothly integrated yet. This could be done by embedding the navigation graph in a small portion of the screen of the search results, including the navigation to "weak synonyms".

Beyond formally authored relationships, statistical methods could also be leveraged to detect relatedness of nodes of GeoSkills. Approaches such as Latent Semantic Analysis, based on the concepts' names or on corpora of mathematical definitions for each of the concepts are likely to create methods to find nodes close-by.

Ideally, such neighbours strategy should also exploit the ontological nature (generalizing Germany's 9. Klasse of Saarland into 9. Klasse of any state of Germany.

Acknowledgement

This research is partially funded by the European Union in the project Open Discovery

Space.

Bibliography

- (Manning) Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008. Available from http://nlp.stanford.edu/IR-book/.
- (i2geoCERME6) Kortenkamp, U., Blessing, A. M., Dohrmann, Ch., Kreis, Y., Libbrecht, P., Mercat, Ch.: Interoperable Interactive Geometry for Europe – First Technological and Educational Results and Future Challenges of the Intergeo Project. (2009); published at CERME 6 - Sixth Conference of European Research in Mathematics Education (FR – Lyon) in the working group 7 http://ife.ens-lyon.fr/editions/editions-electroniques/cerme6/working-group-7
- (I2Geo-Quality) Trgalova, J., Jahn, A. P., Soury-Lavergne, S. "Analyse de ressources pédagogiques pour la géométrie dynamique et évaluation de leur qualité : le projet Intergeo ». Actes du Colloque Espace Mathématique Francophone 2009. Paper
- (GWT) Google Inc., Google Web Toolkit, http://developers.google.com/web-toolkit/ (accessed 2012)
- (GeoSkills-SemWeb-Handbook) Libbrecht, P., Desmoulins, C.: A Cross-curriculum Representation for Handling and Searching Dynamic Geometry Competencies, Handbook of Semantic Web for E-learning, 2009. See http://www.iospress.nl/book/semantic-web-technologies-for-e-learning/
- (OWL) Smith, M., Welty C., McGuinness D., Web Ontology Language, W3C Recommendation, 2004 http://www.w3.org/TR/2004/REC-owl-guide-20040210/
- (CompEd) Libbrecht, P., Desmoulins, C: Comped, a Web-based Competency Ontology Editor for Dynamic Geometry. (2009); published at SWEL'09 @ AIED'09 - Ontologies and Social Semantic Web for Intelligent Educational Systems (UK – Brighton). See http://compsci.wssu.edu/iis/swel/SWEL09/
- (Lucene) http://lucene.apache.org/. Accessed in 2012
- (owlapi) OWLAPI, a library to manipulate OWL ontologies. http://owlapi.sourceforge.net/. Accessed 2009.
- (pellet) Clark & Parsia Inc., Pellet: OWL reasoner for Java. http://clarkparsia.com/pellet/
- (xwiki) XWiki SàRL, XWiki, entreprise wiki for Java. http://www.xwiki.org/
- (XCLAMS) XWiki SàRL and Curriki Inc., the XWiki Collaborative Learning Assets Management System.

OpenMath

Mathematical Computations for Linked Data Applications with OpenMath

Ken Wenzel and Heiner Reinhardt

Fraunhofer-Institute for Machine Tools and Forming Technology IWU,
Chemnitz, Germany
ken.wenzel@iwu.fraunhofer.de

Abstract

Linked data can be used to connect information stemming from usually disparate sources. The Semantic Web combines various kinds of logical reasoning for the inference of additional knowledge from existing data and for consistency checking of data sets. Although mathematics is crucial for most areas where Semantic Web technologies can be applied, support for them is lacking from related standards and tools. This paper presents an approach for integrating OpenMath with RDF data for the representation of mathematical relationships and the integration of mathematical computations into reasoning systems.

1 Introduction

Mathematical relationships are ubiquitous within data sets of elements with numerical properties. The data can be as simple as a list of persons with age and height data or as complex as a mathematical model that uses equations to describe a physical process. Both cases can benefit from using linked data principles [2] to combine multiple data sets or to integrate additional information from external data sources. Besides, there is also already a considerable amount of data available as Linked Open Data[1] that could benefit from mathematical support, e.g. for statistical computations [6].

In our case, we are using linked data principles to connect models of production systems and processes with operating data, e.g. for analyzing the energy performance of production facilities [14]. The related performance indicators (like average energy consumption or energy consumption per part) should be automatically computed under consideration of different calculation models depending on the machine and product types. Another use case are distributed component libraries for process planning of manufacturing operations [4], where descriptions of machines and their components are directly fetched from the web (e.g. from a vendors website), instead of manually building local repositories. The latter is especially interesting if vendors are enabled to supply formulas within their product descriptions to compute properties like power consumption or life-cycle costs for a given operating context.

Although an integration of mathematical computations into linked data environments seems promising, there are only few approaches that try to combine both worlds.

A simple approach was proposed by Sánchez-Macián et al. [11] for extending SWRL with functions for the evaluation of mathematical expressions. They give a rough overview on how OpenMath XML expressions can be embedded into ontologies and used by custom SWRL functions for reasoning. These functions rely on SWRL to collect the parameters required for evaluating the mathematical expressions.

[1] http://linkeddata.org/

A comparable solution could also be realized with SPIN (SPARQL Inferencing Notation)[2] that allows to use SPARQL[3] queries for reasoning and constraint checking over RDF data. The upcoming SPARQL 1.1[4] also supports aggregates, which can be used for, e.g. computing the sum or average of a set of elements. However, SPARQL only support simple mathematical operations and therefore an extension with custom functions would be necessary. Furthermore, expressing rules using SPIN may get complicated for domain experts since mathematical terms have to be nested into SPARQL queries.

Another approach is followed by OntoCAPE [8] and OntoModel [13]. These methods define means for sharing mathematical models in a collaborative environment by using ontologies. While OntoCAPE provides vocabulary for describing equation systems with OWL and RDF [9], OntoModel rather focuses on amending a system of equations expressed in Content MathML with additional meta data. The two approaches are purpose built for representing and solving equation systems with a fixed set of input variables whose values need to be determined, e.g. by a user or a software agent, before running the computation.

A general review on representation of mathematical knowledge on the web, which covers a wide range of possible representation formats and systems, is given by Lange [7]. In conclusion of his review, there are no existing systems available that combine Semantic Web technologies and computer algebra systems for mathematical computations or proof.

Our approach realizes a tight integration between OpenMath and RDF[5] for consuming linked data from computer algebra systems. Complemented by an RDF representation for mathematical expressions it not only allows to publish formulas as linked data but also to extend reasoning systems with inferencing capabilities based on mathematical computations.

In Section 2 an OpenMath content dictionary for retrieval and representation of RDF data and corresponding human-friendly notation are introduced. Section 3 describes an OWL ontology for OpenMath objects that enables serialization as RDF. Based on the presented formalisms a method and architecture for math-enhanced inference is introduced in Section 4.

If not stated otherwise then all examples concerning OpenMath objects are given in the Popcorn[6] [5] OpenMath representation. For reasons of simplicity, we use the publicly accessible Friend of a Friend (FOAF)[7] vocabulary to provide linked data examples in RDF that are presented in this paper by using the Turtle syntax[8].

2 Query RDF data from OpenMath

Existing methods for representing and solving equation systems with RDF and OWL [8, 13] as well as the integration of OpenMath with SWRL [11] use a two-step approach where first the required input values are retrieved and then the computation is explicitly executed using them as parameters. By directly integrating RDF query support into OpenMath, the description of mathematical relationships over linked data and also the reuse of related formulas can be simplified. For this reason, we have created an OpenMath content dictionary for RDF[9] that defines symbols to access linked data from computer algebra systems. The main purpose of this

[2] http://spinrdf.org/
[3] http://www.w3.org/TR/rdf-sparql-query/
[4] http://www.w3.org/TR/sparql11-query/
[5] http://www.w3.org/TR/rdf-concepts/
[6] http://java.symcomp.org/FormalPopcorn.html
[7] http://xmlns.com/foaf/spec/
[8] http://www.w3.org/TR/turtle/
[9] http://www.openmath.org/cd/contrib/cd/rdf.xhtml

CD is the retrieval of RDF resources and their associated properties along with a representation for RDF resources and literals.

2.1 Symbols for RDF retrieval

The following four symbols are provided for retrieving data from an RDF store:

resourceset A unary construction function that takes one string argument that is a Manchester Syntax description[10]. These expressions are a subset of the Manchester Syntax for the definition of OWL classes and restrictions. The result of applying the resourceset function is a set of resource objects.

Example: `rdf.resourceset("foaf:Person and foaf:age some xsd:int[>18]")`

resource An unary construction function with one string argument representing an IRI reference as defined by SPARQL[11]. Possible values are prefixed names in the form `"prefix:resourceName"` or absolute IRIs in the form `"<IRI>"`.

Example: `rdf.resource("<http://example.org/p#Alice>")`

valueset A function to retrieve a set of property values for a specific RDF resource. It takes two arguments where the first argument denotes an RDF property as IRI reference and the second argument an OM object that corresponds to an RDF resource. The result of this function is a set of RDF resources or literal values.

Example:

```
$alice := rdf.resource("<http://example.org/p#Alice>");
rdf.valueset("foaf:knows", $alice)
```

value Works like the valueset function but expects and returns exactly one value. The application of this function results in an error if none or more than one value exists.

Example:

```
rdf.value("foaf:age", rdf.resource("<http://example.org/p#Alice>"))
```

Although it would be possible to directly embed SPARQL queries into OpenMath objects, we haven chosen Manchester Syntax descriptions for querying RDF resources. The main advantage of this approach is type-safety. While SPARQL `SELECT` queries may result in tuples of different cardinality and type (either resource or literal), the result of a Manchester Syntax description is always a set of resources.

Since Manchester Syntax is a concise language for OWL ontologies, it essentially is based on description logic (DL). This is beneficial for fast query evaluation, which can also be directly executed by a DL reasoner, but does not allow the use of variables and hence has limited expressiveness in comparison to SPARQL. Please note that Manchester Syntax descriptions can also be directly translated to SPARQL queries [12] and consequently do not require the use of OWL ontologies for the description of datasets.

We also propose to reuse the vocabulary of the `set1` content dictionary to handle instances of `rdf.resourceset` and `rdf.valueset`. This allows for applying existing `set1` functions like `union`, `intersect`, `in` or `size` for basic set operations.

[10] `http://www.w3.org/TR/owl2-manchester-syntax/#Descriptions`
[11] `http://www.w3.org/TR/sparql11-query/#rIRIref`

2.2 Representation of literal values

The representation of RDF literals in OpenMath is straightforward. Typed RDF literals are mapped according to their datatype to basic OpenMath objects of type integer, IEEE floating point number, character string or byte array. If the literal's type is not directly convertible or should be preserved in OpenMath then the *literal_type* attribute can be used to annotate the corresponding OpenMath object. Plain literals are represented as strings and can be annotated with a language tag by using the attribute *literal_lang*.

2.3 Shortening of IRI references

The functions introduced in Section 2.1 allow the specification of IRI references in a shortened form as so-called prefixed names. For example, the expression

```
rdf.resource("foaf:Person")
```

uses the prefixed name *foaf:Person* that expands to a full IRI by concatenating the namespace IRI referred to by the prefix `"foaf"` with the local part `"Person"` resulting in the full IRI `<http://xmlns.com/foaf/0.1/Person>`.

Mappings between prefixes and their corresponding namespaces can be specified by using the *prefixes* attribute in combination with the *prefix* symbol:

```
rdf.resourceset("foaf:Person and foaf:age some rdfs:Literal"){
  rdf.prefixes -> {
    rdf.prefix("rdfs", "http://www.w3.org/2000/01/rdf-schema#"),
    rdf.prefix("foaf", "http://xmlns.com/foaf/0.1/")
  }
}
```

These prefix declarations are scoped to the annotated OpenMath object including its descendants. Please note that annotations in OpenMath do not automatically propagate to children and therefore an implementor has to take care of the correct propagation of prefix declarations.

2.4 User-friendly input with Popcorn

We have extended the Popcorn notation with additional symbols for concise integration of RDF into OpenMath. These RDF expressions have the form:

```
'@' '@'? PROPERTY_REF? ('(' OM_OBJECT ')' | '[' CLASS_REF ']')
```

Basically, the character `@` introduces references to RDF data. A single `@` as prefix indicates that the result is also only a single value whereas `@@` indicates that the result is a set of zero or more values.

The following examples demonstrate the mapping of symbols from the content dictionary for RDF to compact Popcorn expressions:

```
rdf.resourceset("foaf:Person")              → @@[foaf:Person]
rdf.resource("<http://example.org/Alice>") → @(<http://example.org/Alice>)
$alice := rdf.resource("example:Alice")     → $alice := @(example:Alice)
rdf.valueset("foaf:knows", $alice)          → @@foaf:knows($alice)
rdf.value("foaf:age", $alice)               → @foaf:age($alice)
```

2.5 Integration into computer algebra systems

As with every new content dictionary for OpenMath, the related phrase-books for computer algebra systems (CAS) need to be extended with additional translation rules. Since, to the authors' knowledge, no current CAS supports querying of RDF stores, additional effort for extending existing systems is required.

The implementation of the functions `resource`, `valueset` and `value` is rather simple. The symbol `resource` just defines a special constant and the latter execute basic retrieval operations against an RDF store.

Implementing the symbol `resourceset` needs considerably more work since parsing of Manchester Syntax and transformation to SPARQL (or OWL for DL reasoning) is required. Possible solutions may use extended SPARQL dialects like Terp [12]. Another solution that directly converts the Manchester Syntax descriptions to OWL for leveraging existing DL reasoners is proposed in the following section.

In order to support the `set1` content dictionary, a simple approach would be to first retrieve the data objects and then execute the set operation within the computer algebra system.

However, we suggest a mapping to compound SPARQL expressions that efficiently execute the `set1` operations `union`, `intersect`, `in` or `size` for large-scale RDF data.

3 Representing OpenMath objects as linked data

The content dictionary for RDF as introduced by Section 2 provides sufficient support for integrating linked data with OpenMath. However, a slight weakness of this representation stems from the usage of string constants for IRI references. These make it hard to track referenced classes or properties and to update them if the associated RDF resources are renamed.

A plain RDF encoding of OpenMath objects overcomes these shortcomings and bears additional advantages. Thus, established mechanisms for annotating and linking of RDF data can be used with OpenMath, and furthermore, IRI references and Manchester Syntax descriptions can be directly encoded as RDF (Section 3.2).

3.1 An ontology for OpenMath

The ontology denoted by Figure 1[12] is basically a verbatim mapping of the OpenMath standard to OWL. In contrast to the approach proposed by Robbins [10] for the encoding of Strict Content MathML in RDF, we decided to stay as close as possible to the OpenMath XML encoding to simplify the conversion between both representation formats.

The only difference in comparison to OpenMath XML is that we have removed the `OMR` element that encodes explicit references to other OpenMath objects. These references are unnecessary since RDF itself provides the means for cross-referencing between resources.

3.2 Referencing of RDF data

We propose to use a special encoding for functions from the content dictionary for RDF (Section 2.1). Normally, these functions take string arguments for IRI references or Manchester Syntax descriptions. But when using the RDF encoding of OpenMath objects these arguments can directly be represented as resource IRIs or `owl:Restrictions`.

For example, the OpenMath object

[12]The ontology is available at `http://numerateweb.org/vocab/math`.

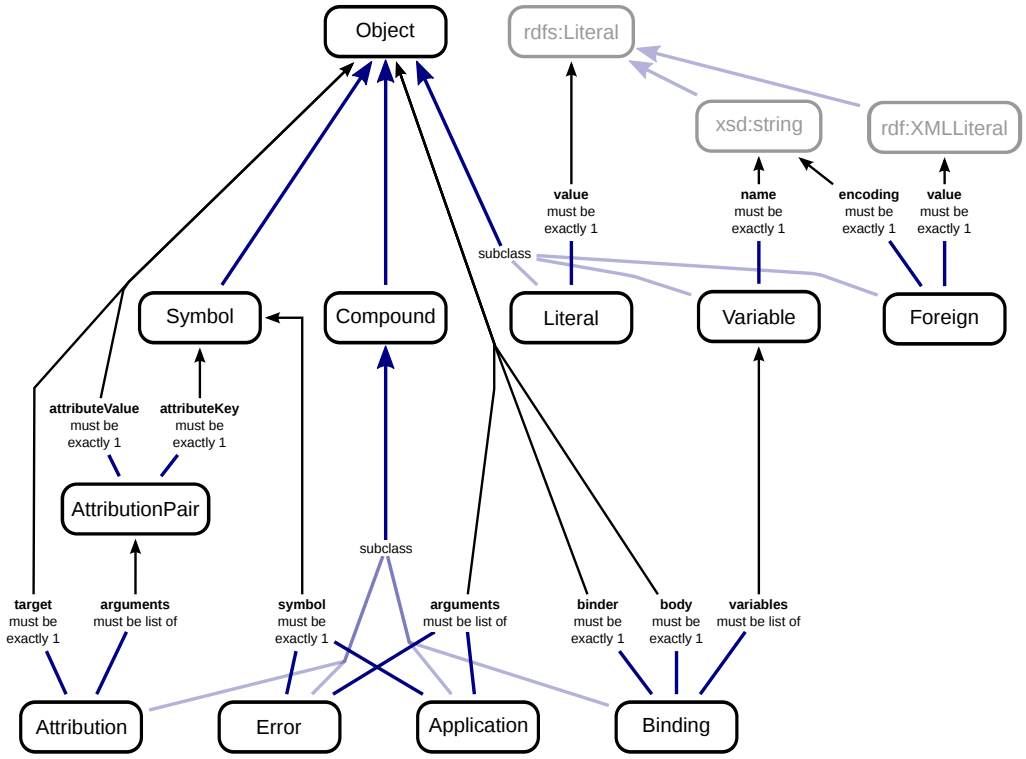


Figure 1: Ontology for OpenMath objects.

```
rdf.resourceset("foaf:Person and foaf:age some xsd:int[>18]")
```

can be encoded in RDF as

```
[ a om:Application; om:symbol <http://www.openmath.org/cd/rdf#resourceset>;
  om:arguments ("foaf:Person and foaf:age some xsd:int[>18]") ] .
```

or when using the native RDF encoding for arguments as

```
[ a om:Application; om:symbol <http://www.openmath.org/cd/rdf#resourceset>;
  om:arguments ([ owl:intersectionOf (
    foaf:Person
    [ a owl:Restriction; owl:onProperty foaf:age;
      owl:someValuesFrom [ a rdfs:DataType; owl:onDataType xsd:int;
        owl:withRestrictions (xsd:minExclusive "18"^^xsd:int) ]
    ]
  )])
] .
```

The latter form represents the Manchester Syntax expression directly as an `owl:Class` description. This encoding allows to use OWL reasoners for determining the contents of the `rdf.resourceset`.

3.3 Extending OpenMath by using RDF and OWL

The ontology as defined in Section 3 enables the representation of OpenMath objects as RDF for sharing mathematical expressions as linked data and for their interoperability with other information serialized as RDF (like `owl:Restrictions`). Together with the content dictionary for RDF as introduced by Section 3.2, it enables the creation of links between OpenMath objects and RDF resources and therefore solves the linking issue mentioned by [6].

Besides this, the ontology does not yet capture the vocabulary for the definition of content dictionaries with symbols and their properties (roles, CMPs, FMPs, etc.). But nonetheless, we can use the ontology to define new OpenMath symbols and relate them to existing ones as discussed by [3]. For example, the following RDF statements define the symbol `asymp1.Landauin` as a subclass of `set1.in`:

```
<http://www.openmath.org/cd/asymp1#Landauin>
  a om:Symbol; rdfs:subClassOf <http://www.openmath.org/cd/set1#in> .
```

In this case, the metamodeling capabilities of OWL 2[13] allow us to treat `asymp1.Landauin`[14] and `set1.in` as OpenMath symbols and as OWL classes at the same time.

The OpenMath ontology can also be extended by additional subclasses of `om:Symbol` (e.g. BinderSymbol, AttributionSymbol, ApplicationSymbol) to enable modeling of symbol roles. These roles can then be applied to symbols by using `rdf:type`:

```
asymp1:Landauin rdf:type om:ApplicationSymbol .
```

If additionally `om:hasCMP` and `om:hasFMP` are introduced as properties of `om:Symbol` then the definition of OpenMath content dictionaries in a distributed way (as linked data) gets possible:

```
set1:emptyset a om:ConstantSymbol;
  om:hasCMP "The intersection of A with the emptyset is the emptyset";
  om:hasFMP [ a om:Application; om:symbol relation1:eq;
    om:arguments (
      # omitted for simplification
    )
  ] .
```

4 Reasoning

Based on the content dictionary for RDF and the OpenMath RDF encoding this section describes how mathematical relationships can be expressed and embedded into OWL ontologies. Furthermore an architecture for respective rule processing is proposed.

4.1 Rule definition

For a human-readable definition of mathematical relationships in an ontology we use the Popcorn syntax and propose the following extension:

```
[Class] : @Property = Expression
```

[13] `http://www.w3.org/TR/owl2-new-features/#Simple_metamodeling_capabilities`
[14] The symbol was introduced as an example by [3].

`Class` specifies a set of individuals in the same manner as an application of `rdf.resourceset` does. Together with `@Property` (essentially an application of `rdf.value`) it defines the domain of the mathematical relationship. `Expression` is any valid Popcorn term including the extensions described in Section 2.4. The rule is to be understood in that way that any instance of `Class` has a `Property` whose value can be calculated by `Expression`.

In order to exemplary describe the reasoning process we use the FOAF vocabulary and extend it with the custom properties `e:mass` and `e:height`. While the former is a measurement for the weight of a person, the latter is used for representing the height of a human being. Further properties are computed using the following rules:

```
[foaf:Person] :
@e:bmi = @e:mass / @e:height ^ 2

[foaf:Group] :
@e:aBMI = sum(@@foaf:member,lambda[$x->@e:bmi($x)]) / set1.size(@@foaf:member)
```

The first rule describes the calculation of the body mass index (`e:bmi`) for any instance of `foaf:Person`. Each individual has to define values for the properties `e:mass` and `e:height`. This can either be done explicitly or by using property values computed by logical reasoning. If we, for example, consider an instance of `foaf:Person` that has a value for `medical:weight` instead of `e:mass` we could use an ontology alignment to infer the required property value:

```
medical:weight rdfs:subPropertyOf e:mass
```

The second rule is used for computing the average body mass index (`e:aBMI`) across a specific set of people. Therefore the sum of respective body-mass-index property values is divided by the cardinality.

4.2 Rule processing

For rule processing we propose the architecture as illustrated in Figure 2. It involves a computer algebra system that has been extended as described in Section 2.5. Thus, this system can handle both, complex mathematics and retrieval of RDF data. In a current implementation of this architecture we make use of Symja[15].

The prototype relies on a simple ontology[16] that is used to represent mathematical rules. It defines the class `mathrl:Constraint` for the representation of such relationships. Instances of `mathrl:Constraint` have the attributes `mathrl:onProperty` for the target RDF property and `mathrl:expression` for the related formula represented by our OpenMath ontology (Section 3). Objects of type `mathrl:Constraint` are assigned to OWL classes by using the property `mathrl:constraint`. The following listing illustrates the serialization of the first rule given in section 4.1.

[15] `http://code.google.com/p/symja/`

[16] The ontology is available at `http://numerateweb.org/vocab/math/rules`.

```
foaf:Person mathrl:constraint [
  mathrl:onProperty e:bmi;
  mathrl:expression [
    a om:Application; om:symbol <http://www.openmath.org/cd/arith1#divide>;
    om:arguments (
      [ a om:Application; om:symbol <http://www.openmath.org/cd/rdf#value>;
        om:arguments (e:mass)]
      [ a om:Application;
        om:symbol <http://www.openmath.org/cd/arith1.xhtml#power>;
        om:arguments ([ a om:Application;
          om:symbol <http://www.openmath.org/cd/rdf#value>;
          om:arguments (e:height)] "2"^^xsd:long)]
    )]
] .
```

During the reasoning process, which can be triggered by the user or on data change, our reasoner iterates over all instances of classes that are related to one or more mathematical rules. Within an iteration it computes the properties given by the `mathrl:onProperty` attribute of the correspondent instances of `mathrl:Constraint` using the mathematical term that is referenced by `mathrl:expression`. Therefore the reasoner uses a phrase-book to transform the mathematical expression for feeding the computer algebra system. The result of the calculation is saved explicitly into the triple store. If, during computation, some of the referenced property values cannot be retrieved then an error is logged. The described process is applicable to backward and forward chaining. Furthermore, it relies on a closed world assumption like SPARQL [1] and thus also SPIN.

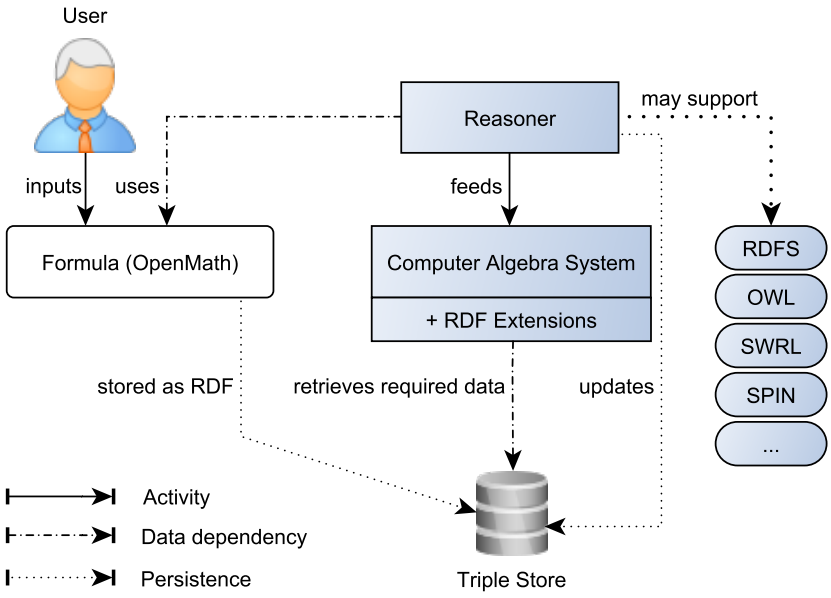


Figure 2: Reasoning architecture for math-enhanced inference.

The examples show that rules may interfere with each other. If the reasoner encounters a rule that is dependent on an individual with a property whose value is retrieved from another rule, then, if not already existent, these respective values are calculated first. Possible strategies to overcome cycles involve the reasoning *until no more changes occur* or *a certain processing limit, e.g. timeout, is reached* or *executing every rule for every individual only once*. So far we can only handle acyclic dependencies.

We additionally support RDFS and OWL-based reasoning, which relies on an open world assumption. The combination with further description logic and rule languages like SPIN or SWRL is possible, but could be a source for additional dependency issues.

5 Conclusion and future work

We have introduced a method for integrating RDF data retrieval into OpenMath to enable the consumption of linked data by computer algebra systems. In combination with our proposed RDF serialization for OpenMath objects it enables the definition of mathematical relationships within OWL ontologies. The related rules are used by our software architecture to execute mathematically enhanced inference over linked data sets.

Future work may investigate how the integration between OpenMath and RDF can be further improved, particularly concerning the representation of RDF statements within OpenMath and the alignment of OWL's set logic with OpenMath's `set1` symbols. The development of methods for partial and distributed calculation could solve scalability issues when applying mathematically enhanced inference on huge data sets.

More sophisticated methods to overcome circular dependencies need to be applied, especially when combining our mathematically enhanced inference approach with reasoning based on description logic or further rule languages.

Finally, we think that, besides our use cases for energy performance analysis and process planning, the integration between OpenMath and linked data has a high potential for many applications, e.g. for open government data where methods for statistical calculations should be transparent. Especially the combination with existing technologies for knowledge representation and logical reasoning may provide improved support for today's engineering tasks (e.g. integration of multiple data sources for the development of sustainable products).

Acknowledgements

The work presented in this paper is co-funded within the Cluster of Excellence Energy-Efficient Product and Process Innovation in Production Engineering (eniPROD®) by the European Union (European Regional Development Fund) and the Free State of Saxony.

References

[1] M. Arenas and J. Pérez. Querying semantic web data with sparql. In *Proceedings of the 30th symposium on Principles of database systems of data*, pages 305–316. ACM, 2011.

[2] Tim Berners-Lee. Linked Data, 2006. `http://www.w3.org/DesignIssues/LinkedData.html`.

[3] James H. Davenport and Paul Libbrecht. The Freedom to Extend OpenMath and its Utility. *Mathematics in Computer Science*, 2(2):253–277, 2008.

[4] B. Denkena, M. Shpitalni, P. Kowalski, G. Molcho, and Y. Zipori. Knowledge management in process planning. *CIRP Annals-Manufacturing Technology*, 56(1):175–180, 2007.

[5] P. Horn and D. Roozemond. OpenMath in SCIEnce: SCSCP and POPCORN. *Intelligent Computer Mathematics*, pages 474–479, 2009.

[6] Christoph Lange. Towards OpenMath Content Dictionaries as Linked Data. In *23rd OpenMath Workshop*, 2010.

[7] Christoph Lange. Ontologies and languages for representing mathematical knowledge on the Semantic Web. *Semantic Web*, 2011.

[8] Jan Morbach, Andreas Wiesner, and Wolfgang Marquardt. OntoCAPE – A (re)usable ontology for computer-aided process engineering. *Computers & Chemical Engineering*, 33(10):1546 – 1556, 2009. Selected Papers from the 18th European Symposium on Computer Aided Process Engineering (ESCAPE-18).

[9] Jan Morbach, Aidong Yang, and Wolfgang Marquardt. OntoCAPE 2.0 – Mathematical Models. Technical Report (LPT-2008-28). Technical report, Lehrstuhl für Prozesstechnik, RWTH Aachen University, 2008. `http://www.avt.rwth-aachen.de/AVT/fileadmin/files/Service_software/Software_Simulation/Mathematical_Models.pdf`.

[10] Andrew Robbins. Semantic MathML, 2009. `http://straymindcough.blogspot.de/2009/06/semantic-mathml.html`.

[11] A. Sánchez-Macián, E. Pastor, J.E.L. De Vergara, and D. López. Extending SWRL to enhance mathematical support. In *Proceedings of the 1st international conference on Web reasoning and rule systems*, pages 358–360. Springer-Verlag, 2007.

[12] E. Sirin, B. Bulka, and M. Smith. Terp: Syntax for OWL-friendly SPARQL queries. In *Proceedings of OWLED*, 2010.

[13] Pradeep Suresh, Girish Joglekar, Shuohuan Hsu, Pavan Akkisetty, Leaelaf Hailemariam, Ankur Jain, Gintaras Reklaitis, Venkat Venkatasubramanian, Bertrand Braunschweig, and Xavier Joulia. OntoMODEL: Ontological mathematical modeling knowledge management. In *18th European Symposium on Computer Aided Process Engineering*, volume Volume 25, pages 985–990. Elsevier, 2008.

[14] Ken Wenzel, Jörg Riegel, Andreas Schlegel, and Matthias Putz. Semantic Web Based Dynamic Energy Analysis and Forecasts in Manufacturing Engineering. In *Glocalized Solutions for Sustainability in Manufacturing*, Life cycle engineering, pages 507–512. Springer-Verlag, Berlin, Heidelberg, 2011.

The GF Mathematical Grammar Library: from OpenMath to natural languages

Olga Caprotti[1] and Jordi Saludes[2] *

[1] Chalmers and University of Gothenburg
caprotti@chalmers.se
[2] Universitat Politècnica de Catalunya
jordi.saludes@upc.edu

1 Introduction

Since 2005 we have been developing a software library for rendering, reading, and translating mathematical expressions, either expressed using formal languages such as OpenMath or LaTeX, or in a number of natural languages. The work begun with the WebALT [1] project as a way to serve mathematical exercises in the native language of the student: in fact the library can be used to generate natural language descriptions of formally encoded mathematical expressions with no loss of meaning. The applications of this technology, coming from the area of grammar-based machine translation are related to the possibility of parsing and generating high quality representations of mathematics.

In this short paper we concentrate on few technical details that made the work interesting from the linguistic point of view. Therefore we introduce the computational linguistic software used as backbone to the work, called Grammatical Framework, and proceed with the presentation of the mathematical library, its organization and modular design. We then discuss some examples that required careful thought.

1.1 The Grammatical Framework

The *Grammatical Framework* (GF) is a type theoretic programming language for writing grammars for multiple languages at once [3]. Multilingual applications use an interlingua: the semantics of an expression in natural language that should be rendered or translated is captured in an *abstract tree*, which is its language-independent representation. As it turns out, the abstract tree representation is also a natural representation of mathematical expressions, one that is also akin to the OpenMath abstract objects.

These trees are described by an *abstract grammar* defining what is possible to express in the specific application, whilst the *concrete grammars* (one for each language) define how the abstract meaning is converted to the given language. Once an abstract grammar is given, to add yet another language to the application amounts to adding a new concrete grammar. Ideally, if a concrete grammar for a language in the same linguistic group is already available, the grammar for the new language is almost an exact copy of the existing grammar, modulo some lexicon adaptations. GF hides all linguistic details of a specific language from the programmer in a low-level *resource grammar library*, so that in principle a domain expert is able to develop new languages for a given application. Details of the GF Grammar Library, including language coverage, are online[1].

*The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° FP7-ICT-247914.

[1] http://www.grammaticalframework.org/lib/doc/synopsis.html

A GF abstract grammar defines how expressions in given *categories* are combined. An example tree in the Mathematical Grammar Library (MGL) looks as follows:

```
mkProp
  (lt_num (abs (plus (BaseValNum (Var2Num x) (Var2Num y))))
  (plus (BaseValNum (abs (Var2Num x)) (abs (Var2Num y)))))
```

When linearized with the English and Spanish concrete grammars, it yields the natural language expressions[2]:

> the absolute value of the sum of x and y is less than the sum of the absolute value of x and the absolute value of y

> el valor absoluto de la suma de x e y es menor que la suma del valor absoluto de x y el valor absoluto de y

As mentioned above, the abstract tree is not far from the OpenMath expression. The linguistic function `mkProp` wraps the wording produced by the subexpressions. In terms of computational linguistic technology, this approach differs from the standard statistical based approaches, such as Google Translate, in that it can generate high quality translations for arbitrarly deep nesting of subexpressions, as opposed to being limited by n-grams distance.

The number of categories on a GF application is a trade-off between how much ambiguity is tolerable and the expressiveness of the whole system. The defined categoriesin the MGL are `Value` X, and `Variable` X where X is a `Number`, a `Function`, a `Set` or a `Tensor` (namely vectors or matrices). The actual version of the library implements these by defining a fixed category for each combination $\{\text{Variable}, \text{Value}\} \times \{\text{Number}, \text{Set}, \text{Function}, \text{Tensor}\}$. Thus, for instance, `VarNum` = `Variable Number` and `ValSet` = `Value Set`. Other categories stand for propositions, geometric constructions and indexes.

Each abstract category corresponds to a linguistic category in a *concrete grammar* of a specific language. Usually a `Value` points to a *noun phrase* and a `Variable` to a string. More complex expressions, those combining categories, correspond in a natural way to linguistic entities composed from these elements: propositions are mapped into *clauses* with grammatical polarity, operations to *sentences* and simple exercises to *texts*.

2 The Mathematical Grammar Library

The library can be organized in a matrix, where the horizontal axis runs over the languages while the vertical axis covers layers of complexity of mathematical expressions.

At present the languages are: Bulgarian, Catalan, English, Finnish, French, German, Hindi, Italian, Polish, Romanian, Russian, Spanish, Swedish and Urdu. As a proof of concept, it includes also a couple of computer software languages which are relevant to mathematics, namely LaTeX and Sage [2].

The vertical axis runs over three layers of increasing complexity:

1. **Ground**: literals, indexes and variables
2. **OpenMath**: modeled after the following *Content Dictionaries*, considered useful for expressing the mathematical fragments at the time of the WebALT project:

[2]Notice the special form of the conjunction "x e y": The usual Spanish conjunction "y" must be changed for euphony before a vowel that sounds alike. It is automatically taken care by the GF Spanish resource grammar.

- arith1, arith2, complex1, integer1, integer2, logic1, nums1, quant1, relation1, rounding1;
- calculus1, fns1, fns2, interval1, limit1, transc1, veccalc1;
- linalg1, linalg2;
- minmax1, plangeo1, s_data1, set1, setname1.

3. **Operations**: takes care of simple mathematical exercises. These appear in drilling exercises and usually begin with directives such as 'Compute', 'Find', 'Prove', 'Give an example of', etc.

Objects in the OpenMath standard [5] relate to GF types, namely each symbol in a Content Dictionary (CD) roughly corresponds to a production of the same name in a GF module named after that CD. Application of functions to numbers are expressed by the production `At` that takes a `Value Function` and a `Value Number` and return a `Value Number`. More examples are in the table 1.

Following the lines of the *Small Type System* [4, principle 4], we imposed that binary associative functions take a list of values and return a value of the same kind. For example, `plus` in `arith1` has signature plus : [ValNum] → ValNum, while the category [ValNum] (meaning a list of numeric values) is declared to take at least two values. Therefore is impossible by construction to add a single number (i. e. "the sum of 3").

3 Linguistic peculiarities

Some interesting points on the implementation are related to language specifics. For example, the simple exercise that asks for computing a numeric value [3]:

```
DoComputeN ComputeV  (determinant (Var2Tensor M))
```

gives in English:

> Compute the determinant of M.

This pattern is shared in most of the languages, so it got abstracted into an *incomplete concrete grammar* file **OperationsI**. From this module, one can get **Operations**L for language L simply by specifiying the lexicon and paradigms modules for this L, in a similar way a function is applied to its arguments. But in French is impolite to use an imperative in this case; Therefore the module **OperationsFre** should re-implement this production in a specific way.

Another point worth mentioning is *function application*. Notice the different forms:

- "the cosine of 3"
- "f at 3"
- "the derivative of the sine at 3"
- "x to the cosine of x where x is 3"

They are all mathematically equivalent but differ in structure: in the first case, the function being applied is a named symbol (the cosine) while in the last one is a λ-abstraction. In the other cases, it is a function variable or it comes from a functional operator.

[3] `determinant` belongs to the OpenMath layer of the library and `Var2Tensor` makes a value out of a variable. `DoComputeN` denotes an exercise asking to compute a number, while `ComputeV` gives finer control on which verb to use to denote computation ('to compute' in this case).

OpenMath	GF
Symbol ***name*** in ***CD***	*name* in module *CD*
Integer n	n converted to `Value` from predefined type `Int` in module `Literals`
Variable ***name***	*name* in category `Variable` X
Application of a on b	a b
Binding $\lambda z\, app$	`lambda` z *app*, where z is a `Variable` and *app* a `Value`.
Attribution, Error, Bytearray	Not supported

Table 1: Some equivalences between the OpenMath standard and grammar library

4 Applications and future development

The library is publicly available at the MOLTO repository [7] and is documented at [8]. It is being used in the *mathbar* demo in the MOLTO project [6] accessible from [9]. An example of natural language interaction with a computer algebra system can be retrieved from the `sage` directory of the library distribution and has been recently presented at [12].

For the future, the library needs to grow in breadth and shape: at this moment, it is systematically tested for three languages but depends on domain experts native speakers to polish the remaining ones.

Integration of natural language productions and formulas is also prominent in the TODO list. This is a variegated issue as [10] shows, but it is necessary for fluent mathematics in applications. Also more natural renderings of logical propositions [11] will open the door to usage in automatic reasoners and theorem provers.

References

[1] `http://webalt.math.helsinki.fi/content/index_eng.html` Last viewed June 2012.

[2] `http://www.sagemath.org/` Last viewed May 2012.

[3] A. Ranta, "Grammatical Framework: programming with Multilingual Grammars," CSLI Studies in Computational Linguistics, Standford, 2011.

[4] J. H. Davenport, "A small OpenMath type system," ACM SIGSAM Bulletin, vol. 34, no. 2, pp. 16–21, Jun. 2000.

[5] OpenMath Consortium, "The OpenMath Standard," OpenMath Deliverable, vol. 1, 2000.

[6] The MOLTO project. `http://www.molto-project.eu/` Last viewed May 2012.

[7] The mathematical library `svn://molto-project.eu/mgl`.

[8] J. Saludes et al., "Simple drill grammar library," `http://www.molto-project.eu/sites/default/files/d61.pdf`. Last viewed May 2012.

[9] Grammatical framework demos. `http://www.grammaticalframework.org/demos/index.html`. Last viewed May 2012.

[10] Mohan Ganesalingam, "The Language of Mathematics." PhD thesis, Cambridge University, 2009.

[11] A. Ranta, "Translating between language and logic: what is easy and what is difficult." Automated Deduction, CADE-23, 2011.

[12] Dominique Archambault, Olga Caprotti, Aarne Ranta and Jordi Saludes, "Using GF in multimodal assistants for mathematics." Digitization and E-Inclusion in Mathematics and Science 2012, Tokyo, Japan.

CICM Work in Progress

An XML-Format for Conjectures in Geometry (Work-in-Progress)

Pedro Quaresma

CISUC/Department of Mathematics, University of Coimbra
3001-454 Coimbra, Portugal, pedro@mat.uc.pt

Abstract. With a large number of software tools dedicated to the visualisation and/or demonstration of properties of geometric constructions and also with the emerging of repositories of geometric constructions, there is a strong need of linking them, and making them and their corpora, widely usable. A common setting for interoperable interactive geometry was already proposed, the I2G format, but, in this format, the conjectures and proofs counterparts are missing. A common format capable of linking all the tools in the field of geometry is missing. In this paper an extension of the I2G format is proposed, this extension is capable of describing not only the geometric constructions but also the geometric conjectures. The integration of this format into the Web-based *GeoThms*, *TGTP* and *Web Geometry Laboratory* systems is also discussed.

1 Introduction

In many dynamic geometry software tools (DGSs), a geometric construction is specified using, explicitly, a formal language. In others, the construction is made interactively, by clicking specific buttons and/or icons, but behind this approach there is also a formal geometric language, although usually hidden from the user. All these languages share many primitive commands (related to geometric constructions), but there are also differences in the set of supported commands, and they follow different syntax rules.

Another important set of tools related to geometric constructions is given by the geometry automated theorem proving software tools (GATPs). Given a geometric construction (eventually created with a given DGS) and a conjecture related to that construction, the GATPs are capable of proving or disproving (although not always) the conjecture. Some of them aim at producing traditional, human readable, geometric proofs [2,5,12].

With a large number of tools focusing on visualising geometric constructions, on proving properties of constructed objects (or both) and repositories of geometric problems (RGPs), there is an emerging need of linking them and making widely usable: constructions; conjectures and proofs generated with different tools. This would help in the progress of the field of geometric constructions, including their role in education.

The I2G format [18] was designed to describe constructions created with a DGS allowing the exchange of geometric constructions between different DGSs.

This format should be complemented in such a way that it can provide support for conjectures. The new format should be a superset of the former format, i.e. a DGS should be able to read the new format, ignoring all the extra information regarding conjectures and proofs. A GATP should be able to read the new format using, if needed, the geometric construction specification. In the following such an extension, the I2GATP format, is discussed.

Some of the most important motivating arguments for using XML in storing descriptions of geometric constructions and conjectures and as an interchange format are: strictly structured files, easy to parse, process, and convert into different forms and formats; a strict content validation of documents with respect to a given set of restrictions; easier communication and exchange of material between unrelated tools.

Paper overview. In Section 2 some background regarding DGSs, the I2G format, GATPs and RGPs is given. In Section 3 the overall structure of the new format is described. In Sections 4 implementations issues are discussed. Finally in Section 5 some final conclusions are drawn and future work is discussed.

2 Background

In this section some basic background information about geometric constructions, the intergeo format (I2G), geometric conjectures and proofs and repositories of geometric problems is given.

2.1 Dynamic Geometry Software

Dynamic geometry software tools (DGSs) allow an easy construction of geometric figures built from free objects, elementary constructions and constructed objects. The dynamic nature of such tools allows its users to manipulate the positions of the free objects in such a way that the constructed objects are also changed, yet preserving the geometric properties of the construction. These manipulations are not formal proofs, as the user is considering only a finite set of concrete positions. Neither the DGS are able to provide a proof of a given conjecture nor they are able to ensure the soundness of the constructions built by its users.

There are multiple DGSs available [1]: GeoGebra, Cinderella, GeometerSketchpad, C.a.R., Cabri, GCLC to name some of the most used.

2.2 Intergeo Format

The Intergeo (I2G) file format is a specification based on the markup language XML designed to describe constructions created with a DGS. It is one of the main

[1] `www.geogebra.org`, `www.cinderella.de`, `www.dynamicgeometry.com/`, `zirkel.sourceforge.net/`, `www.cabri.com/`, `www.emis.de/misc/software/gclc/`

results of the intergeo project, an eContentplus European project dedicated to the sharing of interactive geometry constructions across boundaries. For more information about the project, visit the site `http://i2geo.net` and look into the documentation available there, as well as to [8,9].

An intergeo file takes the form of a compress file package. The main file is `intergeo.xml`, which provides a textual description of the construction in three parts, the elements part describing a (static) initial instance of the configuration, the constraints part where the geometric relationships are expressed and the display part where the details regarding the rendering of the construction are placed. For more details on the file format see [18].

There are already a significant number of DGSs supporting the I2G format (see [3] for details).

2.3 Geometry Automated Theorem Proving

The geometry automated theorem provers (GATPs) give its users the possibility to reason about a given DGS construction, this is no longer a "proof by testing", but an actual formal proof. Another link between the GATPs and the DGSs is given by the automated deductive testing, by the GATP, of the soundness of the constructions made by the DGS [7]. Most, if not all, DGSs are capable of detecting and reporting syntactic and semantic errors, but the verification of the soundness of the construction is beyond their capabilities. If we can link DGSs and GATPs we will be able to use a given GATP in order to check the soundness of a construction created with the help of a DGS.

Automated theorem proving in geometry has two major lines of research: synthetic proof style and algebraic proof style (see [10] for a survey). Algebraic proof style methods are based on reducing geometric properties to algebraic properties expressed in terms of Cartesian coordinates. These methods are usually very efficient, but the proofs they produce do not reflect the geometric nature of the problem and they give only a yes/no conclusion. Synthetic methods attempt to automate traditional geometry proof methods producing human-readable proofs.

If the GATP is capable of producing synthetic proofs, the proof itself can be an object of study, in other cases only the conclusion matters [2,6].

2.4 Repositories of Geometric Problems

When considering repositories of geometric problems we are directly interested in a common format. If we want to provide a repository of geometric problems that can be used by DGSs and GATPs, then the constructions should be kept in a common format that can be converted to the DGS and/or GATP internal format whenever needed. The author of this paper is directly involved in this efforts having three different project that involve repositories of geometric problems.

The first (chronologically) of the mentioned projects is GeoThms[2], a Web-based framework for exploring geometric knowledge integrating DGSs, GATPs,

[2] `http://hilbert.mat.uc.pt/GeoThms/`

and a RGP. The GeoThms is a publicly accessible system with a growing body of geometric constructions and formally proven geometric theorems, its users can easily use/browse through existing geometric contents and build new contents [17]. Within this project a common, XML-based, interchange format for descriptions of geometric constructions, conjectures and proofs was developed [14]. This format predates the I2G format.

A more recent project is the Thousands of Geometric problems for geometric Theorem Provers (*TGTP*)[3]. This is a Web-based library of problems in geometry. *TGTP* aims, in a similar spirit of *TPTP* and other libraries, to provide the automated reasoning in geometry community with a comprehensive and easily accessible library of GATP test problems [15]. The I2GATP format is being developed for this project. For the moment the *TGTP* system still uses the XML-based, interchange format developed for the GeoThms system (the two system share a common database), but it will change to the new format as soon as it becomes stable.

In an educational setting, the project Web Geometry Laboratory (WGL)[4] is an asynchronous/synchronous Web environment that integrates a DGS and a RGP (and it will integrate a GATP in a next version), aiming to provide an adaptative and collaborative blended-learning environment for geometry [19]. Here the need for a common interchange format is less important, nevertheless it will be useful to allow the system to be more easily configurable, i.e. using a common format will allow choosing the DGS and/or the GATP more freely.

2.5 Integration Issues

There are already some systems integrating a DGS with one, or more, GATP and a set of examples (e.g. GCLC [4,6], GeoProof [13], JGEX [1]), but all this systems provide closed tools with a tight integration between different internal functionalities. If we want to be more generic, loosely linking DGSs, GATPs and RGPs, we need a way to establish the communication between tools as unrelated modules, i.e. we need a common format that can be used as a communication channel between tools.

3 Overall Architecture

A common format for geometric constructions, conjectures and proofs should address the communication between DGSs and GATPs, to establish the soundness, by the GATP, of a construction made with the help of the DGS or to prove (or disprove) a given conjecture about a construction made in the DGS:

- The communication between DGSs and GATPs, to establish the soundness, by the GATP, of a construction made with the help of the DGS or to prove (or disprove) a given conjecture about a construction made in the DGS.

[3] `http://hilbert.mat.uc.pt/TGTP`
[4] In prototype stage: `http://hilbert.mat.uc.pt/WebGeometryLab/`

– The rendering of the proof. If the GATP uses an algebraic method only the final result will be usable, but if the GATP uses a synthetic method, the proof itself can be an object of study.

Geometric proofs could appear in many different forms, for instance in axiomatic form (e.g., in Hilbert-style, sequent calculus style, etc.); representing higher-level proofs, produced by the area method; as algebraic proofs produced by the algebraic methods like the Gröbner basis method, etc. The representation of the proof and/or its rendering will always be linked to the method used in its development. This will be addressed in Section 3.2.

3.1 Representation of Constructions, Conjectures and Proofs

In order to enable communication between the geometric tools (i.e DGSs and GATPs) and converting files between different formats a single target format should exist: a format that could define a common normal form for the different tools. The proposal is to extend the I2G format in such a way that the new format would complement the construction description (made by a DGS) with the conjecture description. This new format will be called the I2GATP format.

Converting from a DGS/GATP language to XML, would be performed by a specific converter, naturally relying on the DGS/GATP's parsing mechanism. Converting from XML to a DGS/GATP language, will be implemented via XML-parsing tools.

Having converters from, and to, the I2GATP format for all DGSs and GATPs, we (indirectly) have converters from each tool to any other tool. Thus, in this way, the base for a common interchange format is provided. XML is a natural framework for such interchange format, because of its strict syntax, verification mechanisms, suitable usage on the Internet, and a large number of available supporting tools.

XML descriptions of constructions, conjectures and proofs can be, by means of XSLT, also rendered into other formats that are convenient for human-readable display in browsers. It can also be transformed into different representations, such as natural language form.

A specific XML scheme document could define syntactical restrictions for construction descriptions, conjectures and proofs. This document could then be used, in conjunction with the generic XML validation mechanism, for verifying whether a given file in the I2GATP format is correct (or not).

3.2 Structure of I2GATP Format

Following the ideas of the I2G common format all the files related to the I2GATP format will be packed in a single compressed file, the *container*, which is nothing more then a I2G container with three additional directories. The I2GATP format will be spread in four, at least, XML files (see Figure 1).

Apart from the `intergeo.xml` file, which is mandatory (see the I2G format specification [18]), the other files are optional.

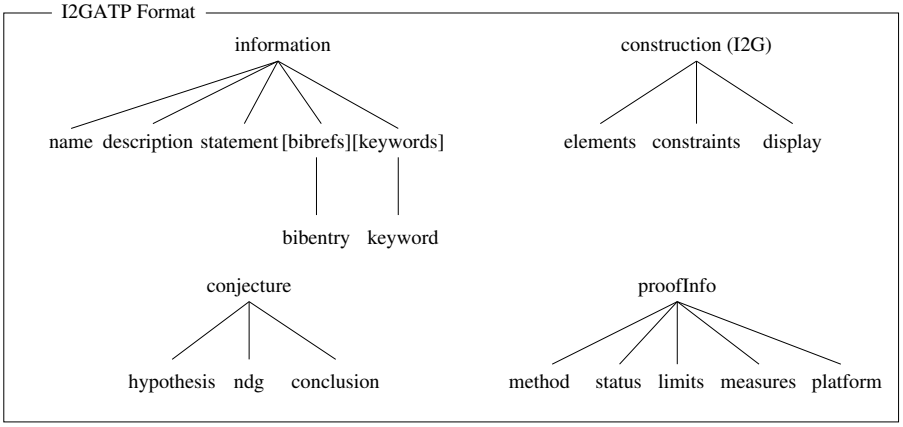


Fig. 1. Structure of the I2GATP File Format

Information The `information.xml` file contains all the generic (human) information about the problem. The *name* of the problem; a brief, informal, *description* of the problem; an informal (rigorous) mathematical description (*statement*) of the problem; a list of bibliographic references; a list of keywords.

Construction The `intergeo.xml` file contains the construction in the I2G format. The I2G format has as main tag the *construction* tag with three sub-nodes: *elements* for the free objects; *constraints* for the objects fixed by construction constraints and *display* for the display details.

Conjecture This is the core of the I2GATP format. In here the *hypothesis*, the *ndg* (non-degenerate conditions) and the *conclusion*, establishing the conjecture to be proved, are specified. The non-degenerate conditions could be a side-effect of the proving process, e.g. automatically generated by a GATP based in the area method, or provided manually.

Proofs For a given problem/conjecture we can have many proof attempts: different approaches, for instance synthetic proof versus algebraic proof; different methods, Gröbner bases method versus Wu's method; different GATPs, GCLCprover versus CoqAM, and all the possible combinations of this three different aspects.

Each proof attempt will be kept in a file `proofInfo.xml` in a sub-directory of the proofs directory (see Section 3.3 for more details).

Each individual proof node will have: the information regarding the GATP, its version and method used; the status of the proof, e.g. *proved*; the computational constraint regarding the proof attempt made by the GATP, e.g. maximum CPU time and RAM space allowed by the system; the proof metrics, e.g. number of proof steps (area method) and the platform used when doing the proof, e.g. CPU, RAM, and other details about the computational platform.

For the proof status the SZS ontology [20] will be used as a base. The "Unsolved" branch will be used as it is, the "Solved" branch has to be adapted to the I2GATP settings.

Given the fact that the proofs produced by different GATPs/Methods are, and should continue to be, quite different we do not try to create a common formats for the proofs. The outcomes produced by the different GATPs will be kept as they are produced (see the *container* in Section 3.3).

3.3 The container

As said above, the I2GATP *container* is a superset of the I2G container, with three additional directories: (`information`; `conjecture` and `proofs`). This means that it will be possible to extract the I2G container out of this file, it will be a simple question of unpacking the file, erasing the additional directories and repacking, if needed, the resulting files.

information/	mandatory
information/information.xml	optional
construction/	mandatory
construction/intergeo.xml	mandatory
construction/preview.pdf	optional
construction/preview.svg	optional
construction/(...)	
conjecture/	mandatory
conjecture/conjecture.xml	optional
proofs/	mandatory
proofs/proof<GATP><Version><Method>/	optional
proofs/proof<GATP><Version><Method>/proofInfo.xml	optional
proofs/proof<GATP><Version><Method>/proofOutput.pdf	optional
proofs/proof<GATP><Version><Method>/(...)	
metadata/	optional
metadata/i2g-lom.xml	optional
resources/	optional
resources/<image_files>	optional
resources/(...)	
private/	optional
private/<domain-name>	optional
private/<domain-name>/<files>	optional

Table 1. The I2GATP container

The structure of the container follows closely the structure of the I2GATP format. The `information`, `construction` and `conjecture` directories will contain the files `information.xml`, `intergeo.xml` and `conjecture.xml` respectively. The directory `construction` may also contain the rendering of the construction in various graphical formats (e.g. PDF, SVG, PNG, etc.).

The directory `proofs` will contain as many sub-directories as proofs attempts were made for the problem in question. The naming convention follows the ideas in the I2G format, that is, after the prefix "proof", the name of the GATP, its version and finally the method used. Given the fact that this is a directory identifier the strings used in these last fields should be conform to the standard naming conventions. In each of this sub-directories the file `proofInfo.xml` will contain the information regarding the proof attempt. This directory may also contain files with the rendering of the proof in different formats (e.g. PDF, HTML, etc.).

The remaining directories follow the structure of the I2G format and can be used to place additional contents produced by the GATPs.

Following the I2G conventions, the suggest naming convention to the container is `problem<problem_name>.zip`.

In the next section the symbol lists, i.e. the tags proposed to this XML-format, are described.

3.4 Symbol Lists

As said above, the container will have "four" (main) XML files: `information.xml`; `intergeo.xml`; `conjecture.xml` and as many `proofInfo.xml` files as proof attempts were made for a given problem. The `intergeo.xml` is described in the I2G common file format, technical report D3.10 [18]. The other three are specific for the I2GATP format and their symbol lists will be described in the next sections.

The symbol lists will be describe in a coarse fashion. For a more detailed account see [16].

Generic Information (`information.xml`) Generic information about the problem. All fields, except the *name*, may be empty.

The tags are: *name*; *description*; *statement*; *bibrefs* and *bibentry*; *keywords* and *keyword*.

The *description* will be a brief, informal, description of the problem in text format and the *statement* will be an informal (rigorous) mathematical description of the problem in MATHML [11].

The *bibrefs* is a list (it may be empty) of bibliographic references in BIBTEXML format[5].

The contents of the *description* and *bibrefs* tags could be automatically converted from LATEX and BIBTEX using, for example, *tex4ht*[6] and BIBTEXML converters respectively.

The *keywords* is a list of keywords in text format. For the moment this field is a free-form text field. For better querying the repositories, an index or a geometric ontology should be considered. Maybe an "open classification", that

[5] `http://bibtexml.sourceforge.net/`

[6] `http://tug.org/applications/tex4ht/`

is, a classification index open to users additions and where the most chosen keywords became, in time, fixed.

Conjecture Information `conjecture.xml` The main tags are: *conjecture*; *hypothesis*, *ndg* (for non-degenerate conditions) and *conclusion*. The three last tags can contain a large number of other tags used to write down the geometric (logical) statements.

Without pretending to be exhaustive we have: *not_equal*; *not_parallel*; *equal*; *plus*; *mult*; *collinear*; *perpendicular*; *parallel*; *midpoint*; *same_length*; *harmonic*; *segment_ratio*. The symbols of the intergeo format regarding the geometric construction can occur here.

Proofs Information `proofInfo.xml` Contains all the information regarding a proof attempt for given problem.

This is a record of the conditions under which the proof was attempted, i.e. the method used (*method*), the limits imposed to the GATP and the computer system used (*limits* and *platform*). Adding to this the proof outcome, i.e. proved, not proved, etc. and also, measures of efficiency, e.g. CPU time used, number of steps, etc. (*status* and *measures*).

In the list of symbols we have (among others): *status*; *limits*; *time_limit_seconds*; *iterations_limit*; *measures*; *CPU_time*; *elimination_steps*; *number_terms_largest_polynomial*; *computer_name*; *clock_speed*; *RAM*; *operating_system*.

4 Implementation

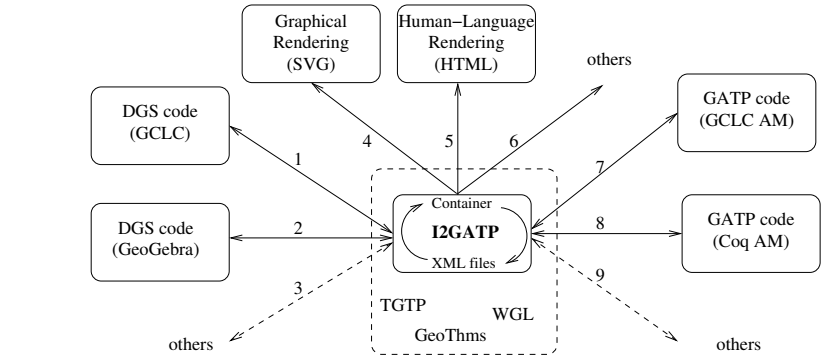
Having defined a XML format for geometric constructions and conjectures its usefulness depends on its support from other tools, i.e. the capability of tools such as DGSs (see [3] to the list of tools already supporting the I2G format) and GATPs to export to the I2GATP format and, of course, its support to other tools in the shape of converters from I2GATP format to the internal format of tools such as the DGSs and GATPs (see Figure 2).

Using the *TGTP* project as a catalyst for this task I will try to provide (working in conjunction with the authors of the tools):

- Converters from dynamic DGSs and GATPs tools (GCL language, Coq AM, etc.) to I2GATP format.
- Converters from I2GATP format to DGSs and GATPs tools (GCL language, Coq AM, etc.).

The I2GATPformat will be backwards compatible with I2G format. DGSs should be able to read the I2GATP container ignoring the extra info. The GATPs should also be able to read the I2G format, adding information whenever needed.

The *TGTP* and *GeoThms* servers will use the I2GATP as its base format, providing converters to and from the different GATPs.



1 – From/to GCLC to/from I2G(ATP)
2 – From/to GeoGebra to/from I2G(ATP)
3 – From/to DGS to/from I2G(ATP)
4 – SVG rendering
5 – HTML rendering
6 – other: proofs; bibrefs., etc.
7 – From/to I2GATP to/from GCLC AM
8 – From/to I2GATP to/from Coq AM
9 – From/to I2GATP to/from GATP

Fig. 2. Conversions From/To I2GATP To/From Geometric Tools

5 Conclusions and Further Work

A case for extending the I2G XML format to the description of geometric conjectures and as an interchange format for dynamic geometry software and geometry automated theorem proving tools was presented.

A brief description of the I2G format and the tools using it and also the tools that can benefit from the extended format was given. The overall architecture and physical organisation of the I2GATP format was described. Arguments justifying the usefulness of this extended format were discussed.

The work presented in this paper is related to work in other domains of automated reasoning where joint efforts of numerous researchers led to standards and libraries which are very fruitful for easier exchange of problems, proofs, and even program code, contributing to the advance of the underlying field (see [15]).

This is a work-in-progress. Questions and future work to be addressed:

- The XML format must be complemented with an extensive set of converters allowing the exchange of information between as many geometric tools as possible.
- The databases queries, as in *TGTP*, raise the question of selecting appropriate keywords. A fine grain index and/or an appropriate geometry ontology should be addressed.
- The I2GATP format does not address proofs. Should we try to create such a format? The GATPs produce proofs in quite different formats, maybe the construction of such unifying format it is not possible and/or desirable in this area.

The I2GATP format will allow to further extend the database of geometric constructions within *GeoThms* and *TGTP* and, hopefully lead then to a ma-

jor public resource for geometric constructions, linking a significant number of geometry tools under this new format.

References

1. Shang-Ching Chou, Xiao-Shan Gao, and Zheng Ye. Java geometry expert. `http://www.cs.wichita.edu/~ye/`, 2004.
2. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, I. multiple and shortest proof generation. *Journal of Automated Reasoning*, 17:325–347, 1996.
3. The Intergeo Consortium. Intergeo implementation table. `http://i2geo.net/xwiki/bin/view/I2GFormat/ImplementationsTable`.
4. Predrag Janičić. GCLC a tool for constructive euclidean geometry and more than that. In Andrés Iglesias and Nobuki Takayama, editors, *Mathematical Software - ICMS 2006*, volume 4151 of *Lecture Notes in Computer Science*, pages 58–73. Springer Berlin / Heidelberg, 2006.
5. Predrag Janičić, Julien Narboux, and Pedro Quaresma. The Area Method: a recapitulation. *Journal of Automated Reasoning*, 48(4):489–532, 2012.
6. Predrag Janičić and Pedro Quaresma. System description: GCLCprover + GeoThms. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 145–150. Springer Berlin / Heidelberg, 2006.
7. Predrag Janičić and Pedro Quaresma. Automatic verification of regular constructions in dynamic geometry systems. In Francisco Botana and Tomás Recio, editors, *Automated Deduction in Geometry*, volume 4869 of *Lecture Notes in Computer Science*, pages 39–51. Springer Berlin / Heidelberg, 2007.
8. U. Kortenkamp, A. M. Blessing, C. Dohrmann, Y. Kreis, P. Libbrecht, and C. Mercat. Interoperable Interactive Geometry for Europe First technological and educational results and future challenges of the Intergeo Project. In *CERME 6*, 2006.
9. U. Kortenkamp, C. Dohrmann, Y. Kreis, C. Dording, P. Libbrecht, and C. Mercat. Using the Intergeo platform for teaching and research. In *Proceedings of the 9th International Conference on Technology in Mathematics Teaching (ICTMT-9)*, 2009.
10. Noboru Matsuda and Kurt Vanlehn. Gramy: A geometry theorem prover capable of construction. *Journal of Automated Reasoning*, 32:3–33, 2004.
11. W3C Math Working Group members. *Mathematical Markup Language (MathML) Version 3.0*. W3C, October 2010.
12. Julien Narboux. A decision procedure for geometry in Coq. *Lecture Notes in Computer Science*, 3223:225–240, 2004.
13. Julien Narboux. A graphical user interface for formal proofs in geometry. *Journal of Automated Reasoning*, 39:161–180, 2007.
14. P. Quaresma, Tomašević J. Janičić, P., M. V.-Janičić, and D. Tošić. *Communicating Mathematics in The Digital Era*, chapter XML-Bases Format for Descriptions of Geometric Constructions and Proofs, pages 183–197. A. K. Peters, Ltd., 2008.
15. Pedro Quaresma. Thousands of geometric problems for geometric theorem provers (tgtp). In Pascal Schreck, Julien Narboux, and Jrgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*, pages 169–181. Springer Berlin / Heidelberg, 2011.

16. Pedro Quaresma. The I2GATP format. Technical report, CISUC, 2012. (`http://hilbert.mat.uc.pt/TGTP/Documents/Docs/cisucTRi2gatp.pdf`).
17. Pedro Quaresma and Predrag Janičić. GeoThms – a Web System for euclidean constructive geometry. *Electronic Notes in Theoretical Computer Science*, 174(2):35 – 48, 2007.
18. E. Santiago, Maxim Hendriks, Yves Kreis, Ulrich Kortenkamp, and Daniel Marquès. I2G Common File Format Final Version. Technical Report D3.10, The Intergeo Consortium, 2010.
19. Vanda Santos and Pedro Quaresma. Integrating DGSs and GATPs in an adaptative and collaborative blended-learning Web-environment. In *First Workshop on CTP Components for Educational Software (THedu'11)*, volume 79 of *EPTCS*, 2012.
20. Geoff Sutcliffe. The SZS ontologies for automated reasoning software. In P Rudnicki, G. Sutcliffe, B. Konev, R. Schmidt, and S. Schulz, editors, *Proceedings of the Combined KEAPPA - IWIL Workshops*, pages 38–49, 2008.

PlanetMath/Planetary

Joseph Corneli[1] and Mircea Alexandru Dumitru[2]

[1] Knowledge Media Institute, The Open University, MK7 6AA, UK
[2] The KWARC Research Group, Jacobs University, D-28759 Bremen, Germany

Abstract. This paper presents our work in progress on the Planetary system, along with a critical evaluation of the project relative to its stated goals and the goals of one of its main "clients", PlanetMath.org.

Keywords: mathematics, learning environments, encyclopedias, Drupal

1 Introduction

The metaphor or analogy implied by the title of this paper is that PlanetMath[3] is like an operating system for mathematics, and Planetary[4] is like a kernel for this system (think GNU/Linux). This analogy is both apt and sloppy.

In particular, PlanetMath is not a complete mathematics "userland". Indeed, it is best known for its mathematics *encyclopedia* which contains over 9000 entries, and defines around 16000 concepts (see [1]). Another key feature of PlanetMath is that every entry is *discussable* via its own attached, threaded, forum. PlanetMath has a variety of other features (like mathematics rendering, a term autolinker, and a workflow and authority model suitable to distributed encyclopedia authoring), most of which were developed in a custom system based on Perl and XSLT (called "Noösphere"), which was written up in Aaron Krowne's 2003 Master's thesis [2]. While this feature set has provided a (mostly) stable and functional basis for a popular community mathematics website for over a decade, the custom nature of the software made extensions and adaptations relatively scarce.

In 2010, the present first author was beginning a Ph. D. project on "Semantic Adaptivity and Social Networking in Personal Learning Environments" that aimed to extend PlanetMath so that encyclopedia entries were "connected to exercises and applications, preliminary materials, and resources for further learning", and to develop software that would track individual performance and provide personalized advice based on aggregated data.

Due to the extensive (and intensive) nature of software modifications that would be required to do this well, he welcomed the possibility to collaborate with Michael Kohlhase and his team at KWARC[5] on a complete re-build of Noösphere, using contemporary web frameworks, and integrating the "KWARC

[3] PlanetMath.org (2001-ongoing), http://planetmath.org
[4] The Planetary System (2010-ongoing), http://trac.mathweb.org/planetary
[5] http://kwarc.info

stack" of semantic technologies into PlanetMath. This led to an early prototype of the Planetary system being named a finalist in Elsevier's Executable Papers Challenge [3], but it was not until this year that an end to the PlanetMath rebuild appeared to be within sight.

The rest of this paper will descibe our technical achievements to date, discuss the immediate road ahead, and reflect on Planetary's potential, particularly from the point of view of its use on PlanetMath. (The reader is referred to [4] for a contemporary high-level overview of Planetary as a whole.)

2 A new "kernel" for math on the web

As indicated above, one of our main goals behind rebuilding PlanetMath's software was to be able to more easily bring new developers into the project. Another was to integrate new technologies.

After our first round of prototyping, Drupal 7 emerged as a good candidate solution for both of these issues. It is a popular system, with a wide variety of contributed modules – and it also supports a healthy marketplace for professional services. So far, the Planetary team has 14 contributors (most of them computer science students at Jacobs University, Bremen), with the current second author focusing on developing Drupal support for features and workflow similar to those found on PlanetMath.

We have found that there are some modules that can be installed and used directly, with minimal configuration (e.g. *privatemsg*, for the exchange of private messages between users) – others needed to be custom-built (e.g. support for *corrections*, essentially a custom form of bug report used to maintain accuracy and quality in PlanetMath's encyclopedia). Some others, like the *userpoints* module can be installed and used with minor tweaks. All in all, we depend on around 25 existing contributed modules, and have written a comparable number of custom modules. For a few legacy features, we took things in a new direction:

1. For mathematics rendering, we are using LaTeXML[6]. Full support for MathML lays the foundation for many other future services. (For example, our Executable Paper demo integrated JOBAD, a Javascript tool for interacting with mathematical documents while reading.)
2. In place of, or alongside, the legacy autolinking service, we have a new interactive ("semi-automated") autolinker, which should provide greater precision for links – and, again, open the door to a range of new interactive services during the document-authoring/editing process [5]. In addition, this feature is made possible by building on top of a real-time collaborative editor Etherpad, so we will get real-time collaboration on mathematics documents "for free".
3. For access to the encyclopedia by Mathematics Subject Classification (MSC), we used a new Linked Open Data (SKOS) implementation of the classification system [6]. This was motivating partly because it allowed us to develop

[6] http://dlmf.nist.gov/LaTeXML/

and demo an integration between Drupal, LaTeXML, and the Virtuoso triple store, which, again, will be useful in a range of future applications (e.g. we will be able to generate RDFa that can then be used to maintain backlinks, for example from an image to all of the articles that include that image).

There are some other features that have been developed for Planetary that are in use in other installation environments (namely, Michael Kohlhase's Computer Science courses), that are not yet integrated into our work on PlanetMath: specifically, an integration with SVN via TNTBase[7], which provides the ability to edit math on the web without ever opening a web browser, and a new books module, which provides support for lengthy documents. These features may eventually make their way into PlanetMath, but they can already serve to illustrate part of our goal in Planetary: to make a system that is useful in many different math-on-the-web contexts, with various features available in the various environments that need them.

One of our core aims in this regard is to make Planetary install "out of the box"; we are currently in the process of using Drupal's *profile* project to package up our work and support this. In future, this should be very helpful both for users and developers.

3 The road ahead

As we can seen in Figure 1, the latest version of Planetary captures many of the same interactions as the legacy version of the site, although the two sites are certainly not identical.

In the short months leading up to CICM in July, we aim to finalize the handful of features that remain unimplemented or partly implemented, and be ready to enter a "beta" with the new platform (i.e., ready to make it the software you see when you browse to PlanetMath.org). From a project management point of view, we have passed our last "milestone" and can now focus on feature-driven development. Our path to deployment on planetmath.org looks like this:

- May 4th: final module tweaking and building for features like requests, private messages, scoring, notices, and the object orphanage.
- May 11th: any final "alpha" features (e.g. finish integrating Etherpad), and a first round of optimizations.
- May 18th: A public "alpha" launch.
- June: Add any "beta" features that we need, in consultation with the PlanetMath user community.
- July: Final improvements readying the site for a switch over to the "beta".

[7] http://tntbase.org/

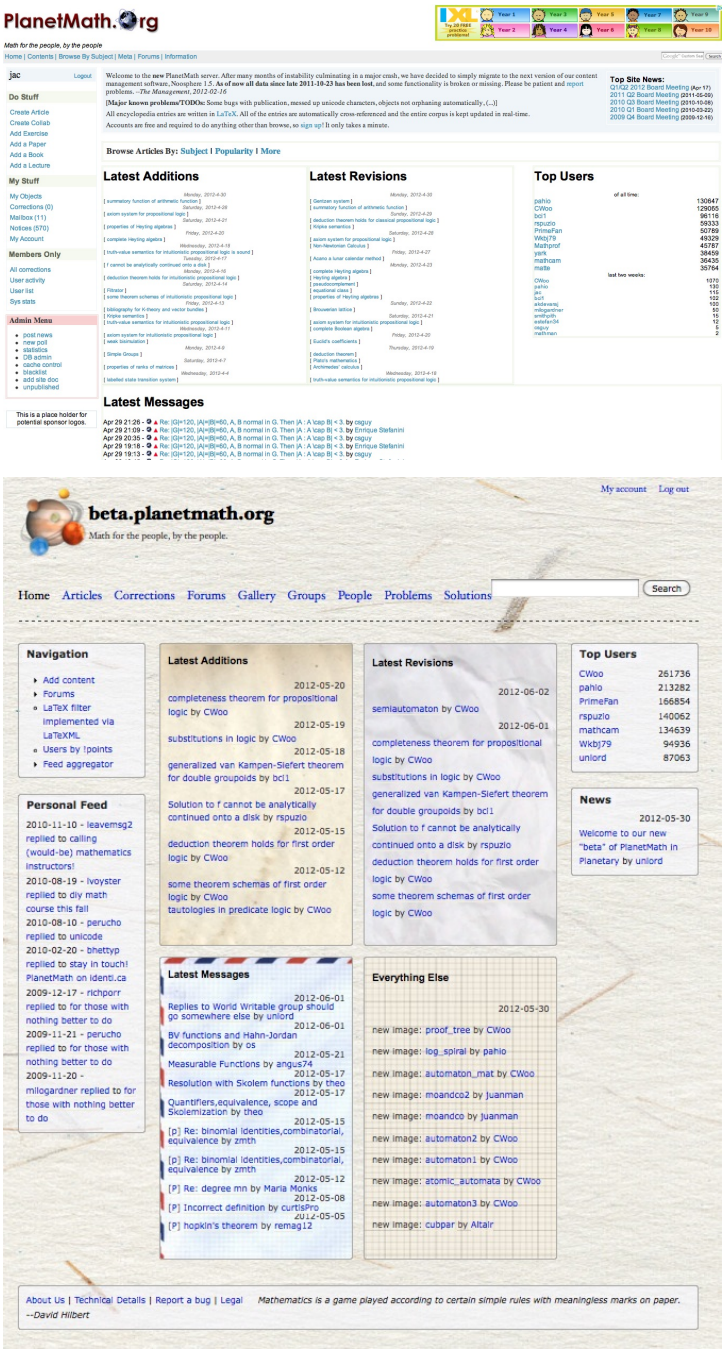


Fig. 1. The current PlanetMath webpage under Noösphere 1.5, and the new "beta" version.

4 A critical evaluation

The question we take up in this section is: is the Planetary system meeting the aims of its developers, and the needs of its user community? What could it do to improve?

Clearly, releasing the system on PlanetMath, and providing links the code[8] and installation instructions[9] should improve things dramatically. The new PlanetMath will render LaTeX blazingly fast, have better links, and a range of new features that address long-standing user concerns and also some nice surprises.

But our aspirations have in fact been much bigger – and switching to Planetary *should* play a big part in bringing them closer. Specifically:

- It is relatively easy to make new content types in Drupal, and we are introducing "problem" and "solution" node types, and allowing people to attach them to encyclopedia articles, and discuss problems with attached "questions" and solutions with attached "reviews".
- Our aim will be to develop some *semantically aware* activity tracking and "heads up" information for people using this system (see Figure 2). It is within reach to provide "related problems" using the MSC classification, but further analysis of theory dependencies (or approximations to the same) should allow us to give links to "simpler related problems", automating a key Pólya heuristic. Even without sophisticated tools, our hope is that a new generation of *students* will feel more encouraged to participate when problems and solutions become "first class" objects in the system.
- Our hypothesis is that the introduction of problems and solutions will provide a vital quality check, and enhancement. In short: encyclopedia articles that do not have attached exercises (or applications) should not necessarily be presumed to be useful. At the same time, exercises that do not have attached solutions may be too hard, i.e., the relevant subjects in the encyclopedia may not be sufficiently developed.

While we are *not there yet*, this is an example of the sort of thing we expect this technology together with the "encyclopedic approach", puts tantalizingly close to within reach.

[8] https://github.com/cdavid/drupal_planetary
[9] https://trac.mathweb.org/planetary/wiki/DrupalPorting

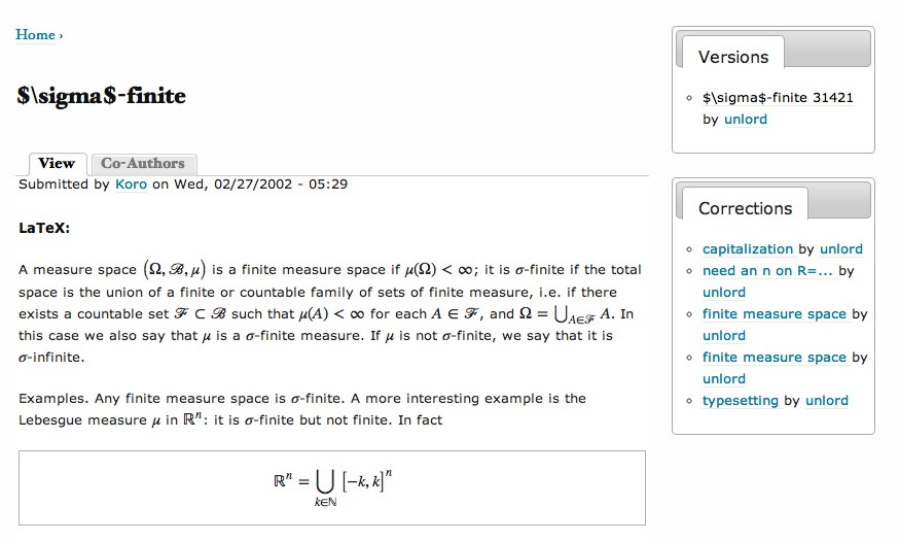


Fig. 2. Important information about articles (like outstanding corrections) in a "heads up" style display, shown here for a recent alpha version.

Note that PlanetMath, unlike, say, Wikipedia, is not *constrained* to be either "just a wiki" or "just an encyclopedia" – so, interactive problem sets and/or peer tutoring are welcome in PlanetMath, though they might not fit so cleanly within the existing Wikimedia family. Rather, the encyclopedic approach envisioned here connects interactions to a carefully curated and *systematic* knowledge base – in contrast with, for example, the StackExchange sites, at least in their current implementation.

In any event, the perspective developed above should brings up some big questions: in brief, what happens to mathematics teaching when students have access to a universal solutions manual for their mathematics course work? We may be able to *measure* whether lecture/homework/test is as effective for learning, as, say, participating in applied research projects.

Still, the hazard here would be to imagine that this can all happen overnight. It has taken PlanetMath 10 years to *define* 16000 terms, how much time will it take to provide a good *exposition* of those terms (always assuming that we do find users who want to participate in this process)?

Furthermore, as we have learned in the last few years, programming Drupal is not equally easy for everyone, documentation is not always clear (or available), and development work is generally a slow process (even with skilled programmers onboard). If the potentially revolutionizing changes (sketched above for mathematics education, but relevant also to research) of math on the web are

to be realized, most aspects of this project will have to scale up a lot – and hopefully coding will be less of a bottleneck.

5 Conclusion

We have described the Planetary system, and discussed its relevance to PlanetMath's continued project of building "a central repository for mathematical knowledge on the web, with a pedagogical slant." We expect the phase of work we will complete this summer to fully renovate and modernize PlanetMath. But once we have readied and deployed an extensible – and re-deployable – core, in a sense, our main work will just be beginning.

For example, we recall the meaning of "planet" from the blogosphere, i.e. planet-as-aggregator. Thinking in this way, PlanetMath might best fulfill its promise not *just* with a great new platform, but by successfully integrating content from other math on the web projects. This sort of aggregation service has yet to be realized, but forms a highly interesting direction for future work.

Indeed, if we are going to do anything about the "$500 million pricetag" for building a math-capable AI[10], we either have to bring about greater efficiencies, or spread the cost out over a relatively large number of people. Without making promises about just when this will be accomplished, we assert that we have, with PlanetMath and now Planetary, taken some vital steps in this direction.

References

1. Joseph Corneli, The PlanetMath Encyclopedia, in *Workshop on Mathematical Wikis (MathWikis-2011) at ITP (2nd International Conference on Interactive Theorem Proving)*, 2011 Nijmegen, Netherlands, August 27th, 2011. (http://ceur-ws.org/Vol-767/paper-03.pdf)
2. Aaron Krowne, An architecture for collaborative math and science digital libraries, Virginia Polytechnic Institute and State University, Masters thesis, 2003.
3. Michael Kohlhase, Joseph Corneli, Catalin David, Deyan Ginev, Constantin Jucovschi, Andrea Kohlhase, Christoph Lange, Bogdan Matican, Stefan Mirea, and Vyacheslav Zholudev, The Planetary System: Web 3.0 & Active Documents for STEM, Proceedings of the International Conference on Computational Science, ICCS 2011, Procedia Computer Science, 4, 2011, pp. 598–607.
4. Michael Kohlhase, The Planetary Project: Towards eMath3.0, in *Intelligent Computer Mathematics, CICM12*, LNAI 6824, J. Davenport, W. Farmer, F. Rabe, and J. Urban, eds., Springer Verlag, 2012.
5. Constantin Jucovschi, Cost-Effective Integration of MKM Semantic Services into Editing Environments, CICM12, 2012.
6. Christoph Lange, Patrick Ion, Anastasia Dimou, Charalampos Bratsas, Joseph Corneli, Wolfram Sperber, Michael Kohlhase and Ioannis Antoniou, Reimplementing the Mathematical Subject Classification (MSC) as a Linked Open Dataset, CICM12, 2012.

[10] http://theconversation.edu.au/if-i-had-a-blank-cheque-id-turn-ibms-watson-into-a-maths-genius-1213

Theorema 2.0: A Graphical User Interface for a Mathematical Assistant System

Wolfgang Windsteiger
RISC, JKU Linz
4232 Hagenberg, Austria

Abstract

Theorema 2.0 stands for a re-design including a complete re-implementation of the *Theorema* system, which was originally designed, developed, and implemented by Bruno Buchberger and his *Theorema* group at RISC. In this paper, we present the first prototype of a graphical user interface (GUI) for the new system. It heavily relies on powerful interactive capabilities introduced in recent releases of the underlying Mathematica system, most importantly the possibility of having dynamic objects connected to interface elements like sliders, menus, check-boxes, radio-buttons and the like. All these features are fully integrated into the Mathematica programming environment and allow the implementation of a modern interface comparable to standard Java-based GUIs.

1 Introduction

Although *Theorema 1.0*, see e.g. [1, 2, 3, 5], has been widely acknowledged as a system with one of the nicer user interfaces, we could observe that outsiders or beginners still had a very hard time to successfully use the *Theorema* system. This was true for entering formulae correctly as well as for proving theorems or performing computations. *Theorema 1.0* as well as *Theorema 2.0* are implemented on top of Mathematica, one of the leading computer algebra systems developed by Wolfram Research. Thus, the principal user interface to *Theorema* is given by the Mathematica notebook front-end. While the 2D-syntax for mathematical formulae available since Mathematica 3, see [6], is nice to read, a wrongly entered 2D-structure has always been a common source of errors. More than that, the user-interaction pattern in *Theorema 1.0* was the standard 'command-evaluate' known from Mathematica, meaning that every action in *Theorema 1.0* was triggered by the evaluation of a certain *Theorema* command implemented as a Mathematica program. As an example, giving a definition meant evaluation of a Definition[...]-command, stating a theorem meant evaluation of a Theorem[...]-command, proving a theorem meant evaluation of a Prove[...]-command, and performing a computation meant evaluation of a Compute[...]-command. For the new *Theorema 2.0* system, we envisage a more 'point-and-click'-like interface as one is used to from modern software tools like an emailing-environment or office software.

The main target user-group for *Theorema* are mathematicians, who want to engage in formalization of mathematics or who just want to have some computer-support in their proofs. Also for students of mathematics or computer science and for teachers at universities or high schools the system should be a tool helping to grasp the nature of proving. Therefore, nice two-dimensional input and output of formulae in an appearance like typeset or handwritten mathematics is an important feature. On the other hand, the unambiguous parsing of mathematical notation is non-trivial already in 1D, supporting 2D-notations introduces some additional difficulties.

Theorema is a multi-method system, i.e. it offers many different proving methods specialized for the proof task to be carried out. The main focus is mainly on a resulting proof that comes as close as possible to a proof done by a well-educated mathematician. This results in a multitude of methods, each of them having a multitude of options to fine-tune the behaviour of the provers.

This is on the one hand powerful and gives many possibilities for system insiders, who know all the tricks and all the options including the effect they will have in a particular example. For newcomers, on the other hand, the right pick of an appropriate method and a clever choice of option settings is often an insurmountable hurdle. The user interface in *Theorema 2.0* should make these selections easier for the user. Furthermore, there should be the possibility to extend the system by user-defined reasoning rules and strategies.

Finally, the integration of proving, computing, and solving *in one system* will stay a major focus also in *Theorema 2.0*. Compared to *Theorema 1.0*, the separation between *Theorema* and the underlying Mathematica system is even stricter, but the integration of Mathematica's computational facilities into the *Theorema* language has improved.

Some of the features described in this paper rely or depend on their implementation in Mathematica. This requires a certain knowledge of the principles of Mathematica's programming language and user front-end in order to understand all details given below. The rest of the paper is structured as follows: the first section describes the new features in recent releases of Mathematica that form the basis for new developments in *Theorema 2.0*, in the second section we introduce the new *Theorema* user interface, and in the conclusion we give a perspective for future developments.

2 New in Recent Versions of Mathematica

We describe some of the new developments in recent Mathematica releases that were crucial in the development of *Theorema 2.0*.

2.1 Mathematica Dynamic Objects

Typical graphical user interfaces nowadays are implemented in the Java programming language and its derivations or extensions. Earlier versions of Mathematica offered the so-called *GUIKit* extension, which was based on Java and used MathLink for communication between Mathematica and the generated GUI. We used GUIKit earlier for the development of an educational front-end for *Theorema*, see [4], but the resulting GUI was cumbersome to program, unstable, and slow in responding to user interaction. As of Mathematica version 6, and then reliably in version 7, see [7], the concept of *dynamic expressions* was introduced into the Mathematica programming language and fully integrated into the notebook front-end. Dynamic expressions form the basis for interactive system components, thus, they are *the* elementary ingredient for the new *Theorema 2.0* GUI.

In short, every Mathematica expression can be turned into a dynamic object by wrapping it into `Dynamic`. As the most basic example, `Dynamic[`*expr*`]` produces an object in the Mathematica front-end that displays as *expr* and automatically updates as soon as the value of one of the parameters, on which *expr* depends, changes. In addition, typical interface elements such as sliders, menus, check-boxes, radio-buttons, and the like are available. On the one hand, the appearance of these elements depends on values of variables connected to them. On the other hand, every action performed on them, e.g. clicking a check-box or radio-button, changes the value of the respective variable. The set of available GUI objects is very rich and there is a wide variety of options and auxiliary functions in order to influence their behaviour and interactions. These features allow the construction of arbitrarily complicated dynamic interfaces and seem to constitute a perfect platform for the implementation of an interface to the *Theorema* system. A big advantage of this approach is that the entire interface programming can be done inside the

Mathematica environment, which in particular brings us a uniform interface on all platforms from Linux over Mac until Windows for free.

2.2 Cascading Stylesheets

Stylesheets are a means for defining the appearance of Mathematica notebook documents very similar to how stylesheets work in HTML or word processing programs. The mere existence of a stylesheet mechanism for Mathematica notebooks is not new, but what is new since version 6 is that stylesheets are cascading, i.e. stylesheets may depend on each other and may inherit properties from their underlying styles just like CSS in HTML. This of course facilitates the design of different styles for different purposes without useless duplication of code. The more important news is that stylesheets can now, in addition to influencing the appearance of a cell in a notebook, also influence the *behaviour* of a cell. This is a feature that we always desired since the beginning of *Theorema*: an action in Mathematica is always connected in some way to the evaluation of a cell in a notebook, and we wanted to have different evaluation behaviour depending on whether we want to e.g. prove something, do a computation, enter a formula, or execute an algorithm. Using a stylesheet, we can now define computation-cells or formula-cells, and the stylesheet defines commands for their pre-processing, evaluation, and post-processing.

Cascading is a nice feature for maintenance of stylesheets also, because is allows to separate settings responsible for behaviour from those for appearance. This is convenient for a system user, who typically would never wish to influence behaviour, because the functioning of the system relies on proper settings in this area. Still, adding new styles for different tastes and occasions such as presentations or lecture notes can be added with ease.

3 The *Theorema* Interface

As said, the Mathematica notebook front-end is the primary user interface for *Theorema*. "Working in *Theorema*" consists of *activities* that themselves require certain *actions*. As an example, a typical activity would be "to prove a formula", which requires actions such as "selecting a proof goal", "composing the knowledge base", "choosing the inference rules and a proof strategy", etc. The central new component in *Theorema 2.0* is the *Theorema commander*; it is the GUI component that guides and supports all activities. Of course, most activities work on mathematical formulae in one or the other way. Formulae appear as definitions, theorems or similar containers and are just written into Mathematica/*Theorema* notebook documents that use one of the *Theorema* stylesheets. We call the collection of all available formulae the *Theorema environment*. Composing and manipulating the environment is just another activity and therefore supported from the *Theorema* commander. The second new interface component in *Theorema 2.0* is the *virtual keyboard*; its task is to facilitate the input of math expressions, in particular 2D-input. Figure 1 shows a screen shot of *Theorema 2.0* with a *Theorema*-styled notebook top-left, the *Theorema* commander to its right, and the virtual keyboard underneath.

3.1 The *Theorema* Environment

The *Theorema* environment is composed in *Theorema*-styled Mathematica notebooks, which have all the capabilities of normal Mathematica notebooks plus the possibility to process expressions in *Theorema* language inside environment cells. This means that *Theorema* expressions are embedded in a full-fledged document format for mathematical writing. Mathematica notebooks consists of hierarchically arranged cells, whose nesting is visualized with cell brackets on

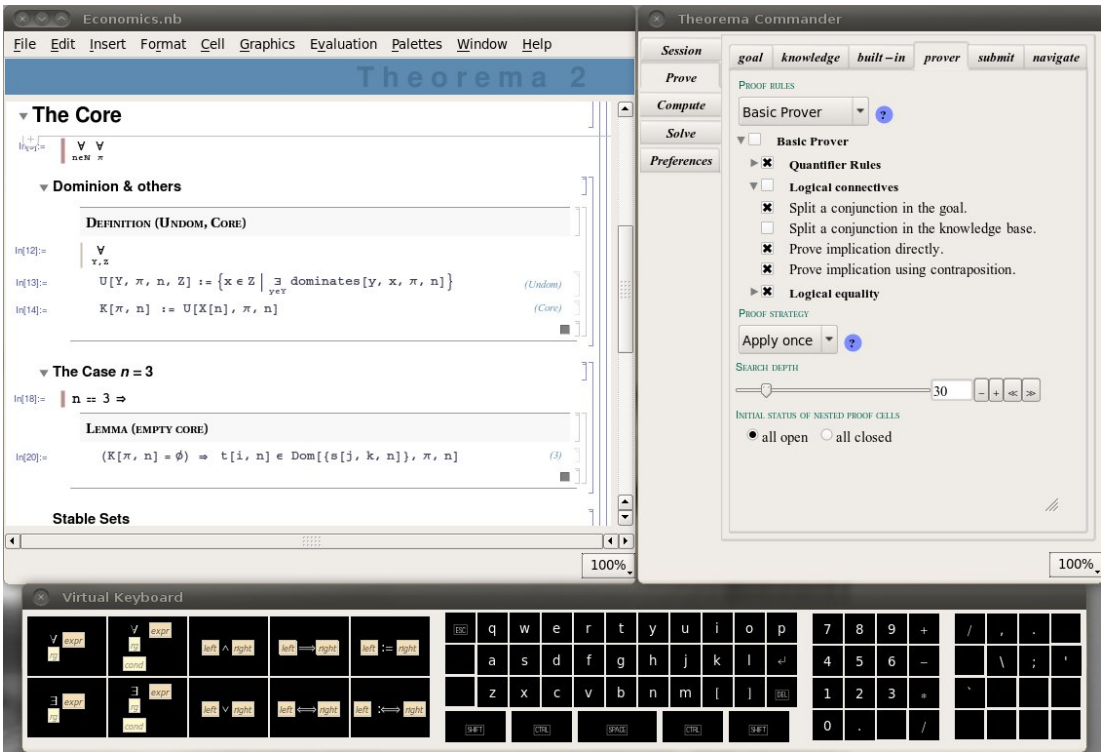


Figure 1: The *Theorema 2.0* GUI

the right margin of the notebook. Figure 1 shows a notebook using a stylesheet that renders the cell brackets with thin blue lines and displays section headings with a small open/close icon to their left for quick opening and closing entire section blocks. Note in particular that each environment forms a group for its own.

Environment cells contain mathematical expressions in *Theorema* syntax with an additional label. If no label is given by the user, an incremental numerical label is automatically assigned. If a chosen label is not unique within a notebook, the user is warned but uniqueness is not enforced. Invisible for the user, the formula is stored in the *Theorema* environment using a datastructure that carries a *unique key* for each formula consisting of the absolute pathname of the file, in which it was given, and the unique cell-ID in that notebook, which is provided by the Mathematica front-end. The formula key allows to uniquely reference each formula in the current environment. As we will explain later, the user never sees nor needs the concrete formula key explicitly.

In mathematical practice, universal quantification of formulae and conditioning is often done on a global level. As an example take definitions, which often start with a phrase like "Let $n \in \mathbb{N}$. We then define ...", which in effect expresses a universal quantifier for n plus the condition $n \in \mathbb{N}$ for all notions introduced in the current definition. For this purpose, we provide *declaration cells*, which may either contain one ore several "orphaned" universal quantifiers (each containing a variable and an optional condition, but missing the formula, to which they refer) or an "orphaned" implication (missing its right hand side). The idea is that the scope of these quantifiers or implications ranges from their location in the notebook to the

end of the nearest enclosing cell group. In the example in Figure 1, this is used in DEFINITION (UNDOM, CORE) with a universal quantifier for Y and Z valid for both formulae inside this definition. The cell grouping defined in the stylesheet ensures that a definition gets its own cell group that limits the scope of the quantifier.

We generalized the idea of declarations inside an environment towards declarations inside an arbitrary cell group. This has the effect that a declaration cell can be put anywhere in a notebook, and its scope ranges as described above from its position to the end of the nearest enclosing cell group. In Figure 1, this is used twice:

1. There is a '$\underset{n\in\mathbb{N}}{\forall}\ \underset{\pi}{\forall}$' at the beginning of Section 'The Core'. This means, that, without further mentioning, n and π are universally quantified with an additional condition $n \in \mathbb{N}$ in the entire section including all its subsections.

2. There is a '$n = 3 \Rightarrow$' in Subsection 'The Case $n = 3$', so that this condition on n affects only in this subsection.

At the moment of giving a formula to the system, i.e. evaluating the environment cell in Mathematica, all declarations valid at this position are silently applied and the actual formula in the *Theorema* environment has all respective quantifiers and implications attached to it just as if they were written explicitly with each formula. This comes very close to how mathematicians are used to write down things and this is very convenient. For bigger documents, one might loose the overview on which declarations are valid at some point. The *Theorema* commander gives some assistance in this situation: by just pressing a button one can obtain a list of all declarations valid at the current cursor position in the selected notebook. Also, you can always view the entire *Theorema* environment (with all formulae currently available including all quantifiers and conditions) from the *Theorema* commander.

3.2 The *Theorema* Commander

The *Theorema* commander, see Figure 1 top-right, is the main GUI component in current *Theorema 2.0*. It is a two-level tabview with activities on the first level and the corresponding actions for each activity on the second level. The first-level activity-tabs can be accessed through the vertical tabs on the left margin. Currently, the supported activities are 'Session', i.e. working on the *Theorema* environment, 'Prove', 'Compute', 'Solve', and 'Preferences'. As the system develops, this list may increase. For each of these activities, the respective actions can be accessed via the horizontal tabs on top. Moving through them from left to right corresponds to a wizard guiding the user through the respective activity. Proving is presumably the most involved activity and we will describe some ideas for its support in the next paragraph in more detail. The remaining parts of the *Theorema* commander are of similar fashion, we will only mention some highlights in the concluding paragraph of this section.

The 'Prove'-activity The example in Figure 1 displays the 'Prove'-tab. It shows actions such as 'goal', 'knowledge', etc. that just correspond to the actions required for proving a formula in *Theorema*, namely defining the proof goal, specifying the knowledge available in the proof, setting up built-in knowledge, and selecting the desired prover to be used. Defining the proof goal is as simple as just selecting a formula in an open notebook with the mouse. The selected formula is shown in the 'goal'-tab, and it changes with every mouse selection. Finally, the choice is confirmed by just pressing a button in the 'goal'-tab. From this moment on, whatever the mouse selects, the proof goal is fixed until the next confirmation.

Goal confirmation automatically proceeds to the tab for composing the knowledge base, see Figure 2 (left). The *knowledge browser* displays a tab for each open notebook or loaded knowledge archive[1]. In each tab, a hierarchical overview of the file/archive content showing only the section structure, environments, and formula labels is displayed. Simply moving the mouse cursor over the label opens a tooltip displaying the whole formula, clicking the label jumps to the respective position in the corresponding notebook/archive. Each entry in the browser has a check-box attached to its left responsible for toggling the selection of the respective unit. In this way, individual formulae, environments, sections, up to entire notebooks can be selected or deselected with just one mouse-click, and the formulae selected in this way constitute the knowledge base for the next prove call. The formula label displayed in the browser is only syntactic sugar, the check-box is connected to the unique key of each formula in the environment, see Section 3.1.

The next action within the 'Prove'-activity is the selection of built-in knowledge[2], see Figure 2 (right). The *built-in browser* works like the knowledge browser described above. Instead of section grouping we have (not necessarily disjoint) thematic groups of built-ins like sets, arithmetic, or logic.

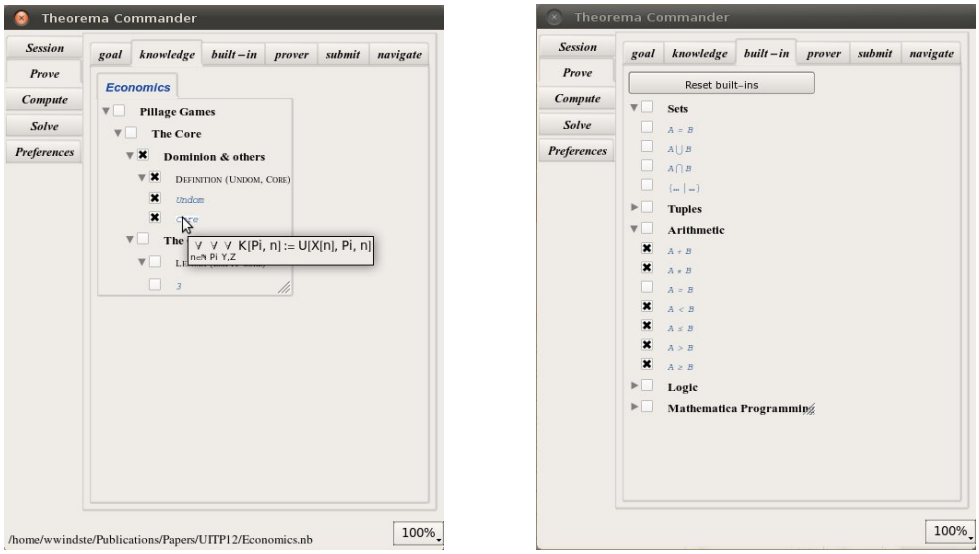


Figure 2: Graphical support for the 'Prove'-activity: the knowledge browser (left) and the built-in browser (right).

After having composed the relevant built-in knowledge, the user needs to select the prover. A *prover* in *Theorema 2.0* consists of a structured list of inference rules accompanied with a prove strategy. Accordingly, the 'prover'-tab, see Figure 3 (left), shows menus for choosing the inference rules and the strategy, respectively, together with short info panels explaining the current choice. A list of inference rules is a list, whose entries are individual inference rules or

[1] *Archives* are another new development in *Theorema 2.0*. An archive gives the possibility to store the formulae from a notebook efficiently in an external file, such that they can be loaded quickly into a *Theorema* session. We do not go into further details in this paper.

[2] With *built-in knowledge* we refer to knowledge built into the *Theorema* language semantics. As an example, '+' is by default an uninterpreted operator. Using some built-in knowledge one can link '+' to the addition of numbers available in the *Theorema* language. This is a feature inherited from *Theorema 1.0*.

themselves lists of inference rules. In addition, every list of inference rules has a name. There is no limit to the nesting of inference rule lists. The 'prove'-tab displays an *inference rule browser* corresponding to the selected rule list, which works like the knowledge browser described above using the rule list structure for the hierarchy and the list names instead of section titles. With the inference rule browser the user can efficiently deactivate individual inference rules, e.g. for influencing whether an implication will be proved directly or via contraposition. In addition, some options for the prover can be set or adjusted from this tab.

The next step is submitting the proof task. The respective tab collects all settings from the previous actions, in particular the chosen goal and knowledge base, and displays them for a final check. Hitting the 'Prove'-button submits all data to the *Theorema* kernel and proceeds to the 'navigate'-tab, see Figure 3 (right), which displays the corresponding proof tree as it develops during proof generation. The nodes in the proof tree differ in shape, color, and content depending on node type and status. As soon as the proof is finished, some proof information is written back into the notebook, in which the proof goal is defined. In addition to an indicator of proof success or failure and a summary of settings used at the time of proof generation, this information contains two important buttons:

1. A button to display the proof in natural language in a separate window. This feature is in essence the same as we had it in *Theorema 1.0*, see e.g. [5]. The 'navigate'-tab in the *Theorema* commander is connected to the proof display in that all labels in the proof tree representation are hyperlinks to the respective text blocks in the proof display describing the corresponding proof step, which is a nice possibility to navigate through a proof.

2. A button to restore all settings in the *Theorema* commander to the values they had at the time of proof generation, which is a quick way to rerun a proof.

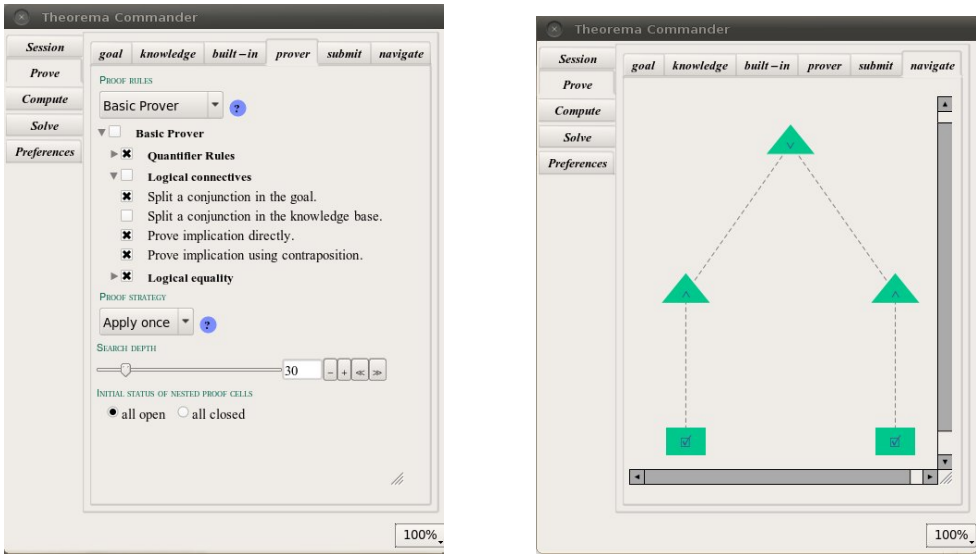


Figure 3: Graphical support for the 'Prove'-activity: the 'prover'-tab (left) and the 'navigation'-tab (right).

Other activities The 'Session'-activity consists of structuring formulae into definitions, theorems, etc., arranging global declarations, inspecting the environment, inputting formulae, and the development and maintenance of knowledge archives. The 'Compute'-activity contains setting up the expression to be computed, defining the knowledge base, and defining the built-in knowledge using knowledge- and built-in browsers as described for proving above. Knowledge selections for proving are independent from those used for computations. In the 'Preferences'-activity we collect everything regarding system setup, such as e.g. the preferred language. The entire GUI is language independent in the sense that no single english string (for GUI labels, button labels, explanations, tooltips, etc.) is hardcoded in its implementation, but all strings are constants, whose definitions are collected in several language-setup files. For a translation to a new language, only these files have to be copied and the english texts in them translated. The language selection menu in the 'Preferences' will immediately offer the new choice for the language, the user selects it, and voilà the GUI runs in the new language. Language support is important in particular for educational purposes that we envisage for *Theorema 2.0*.

An important detail that makes this approach possible is the decision to make the source code available under GPL license. This gives all users access to the source code and in particular the language-setup files. An attractive perspective for user contribution to the system could also be the development of new proof strategies. They are just Mathematica programs applying inference rules, and there is a rich library of *Theorema* programs that is ready for use in the implementation of strategies.

3.3 The Virtual Keyboard

The last component to be described briefly is the *virtual keyboard*, see the screenshot in Figure 1. Although much input can be given through buttons and palettes, such as buttons for frequently used expressions in the 'Session'-tab or the built-in Mathematica palettes, symbols or digits in an expression are most conveniently typed directly on the keyboard. When working with *Theorema 2.0* on a tablet computer or on an interactive white-board, however, e.g. in an educational context, we have no physical keyboard available. For situations like this we provide the virtual keyboard, which is an arrangement of buttons imitating a physical keyboard. It consists of a character block for the usual letters and a numeric keypad (numpad) for digits and common arithmetic operators like on common keyboards. As a generalization of the numpad, we provide a *sympad* (to the far right) and an *expad* (to the left) for common mathematical symbols and expressions, respectively. Using modifier keys like Shift, Mod, Ctrl and more, every key on the board can be equipped with many different meanings depending on the setting of the modifiers. We believe that the virtual keyboard is a very powerful input component for mathematical expressions, which will prove useful even in the presence of a physical keyboard.

4 Conclusion

Theorema 2.0 is currently under development. The components described in this paper are all implemented and the screenshots provided show a running and working system, it is not the sketch of a design. However, the interface presented here is incomplete and it will grow with new demands. From the experience with Mathematica's GUI components gathered up to now we are confident that all requirements for a modern interface to a mathematical assistant system can easily be fulfilled based on that platform.

Some of the features are implemented currently as 'proof of concept' and need to be completed in the near future to get a system that can be used for case studies. As an example,

the *Theorema* language syntax, from parsing via formatted output to computational semantics, is only implemented for a fraction of what we already had in *Theorema 1.0*. Due to the fact that the already implemented parts are the most complicated ones and that we paid a lot of attention to a generic programming style, there is hope that progress can be made quickly in that direction.

The bigger part of the work to be done is the re-implementation of all provers that we already had in *Theorema 1.0*. What we already have now is the generic proof search procedure and the mechanism of inference rule lists and strategies with their interplay. Two sample strategies, one that models more or less the strategy used in *Theorema 1.0* and another one that does a more fine-grained branching on alternative inference rules being applicable, are already available, but no report on their performance can be given at this stage. The big effort is now to provide all the inference rules for standard predicate logic including all the extensions that the *Theorema* language supports. As soon as this is completed we can engage in case studies trying out the system in some real-world theory formalization and in education, for which we plan a hybrid interactive-automatic proof strategy to be available.

References

[1] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards Computer-Aided Mathematical Theory Exploration. *Journal of Applied Logic*, 4(4):470–504, 2006.

[2] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, and W. Windsteiger. The Theorema Project: A Progress Report. In M. Kerber and M. Kohlhase, editors, *Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning)*, pages 98–113. St. Andrews, Scotland, Copyright: A.K. Peters, Natick, Massachusetts, 6-7 August 2000.

[3] B. Buchberger, T. Jebelean, F. Kriftner, M. Marin, E. Tomuta, and D. Vasaru. A Survey of the Theorema project. In W. Kuechlin, editor, *Proceedings of ISSAC'97 (International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii, July 21-23, 1997)*, pages 384–391. ACM Press, 1997.

[4] G. Mayrhofer, S. Saminger, and W. Windsteiger. CreaComp: Experimental Formal Mathematics for the Classroom. In Shangzhi Li, Dongming Wang, , and Jing-Zhong Zhang, editors, *Symbolic Computation and Education*, pages 94–114, Singapore, New Jersey, 2007. World Scientific Publishing Co.

[5] W. Windsteiger, B. Buchberger, and M. Rosenkranz. Theorema. In Freek Wiedijk, editor, *The Seventeen Provers of the World*, volume 3600 of *LNAI*, pages 96–107. Springer Berlin Heidelberg New York, 2006.

[6] S. Wolfram. *The Mathematica Book*. Wolfram Media/Cambridge University Press, third edition, 1996.

[7] Wolfram Research, Inc. Mathematica edition: Version 7.0, 2008. Champaign, Illinois.

JBIG2 Supported by OCR

Radim Hatlapatka

Masaryk University, Faculty of Informatics,
Botanická 68a, 602 00 Brno, Czech Republic
208155@mail.muni.cz

Abstract. Digital Mathematical libraries contain a large volume of PDF documents containing scanned text. In this paper we describe how this documents can be compressed and thus provide them more effectively to the users. We introduce a JBIG2 standard for compressing bitonal images such as scanned text and we discuss issues if OCR is used for improving the compression ratio of jbig2enc open-source encoder. For this purpose we have designed API for using OCR in jbig2enc which we describe in this paper together with already achieved results.

Keywords: `jbig2enc`, JBIG2, PDF size optimization, compression, DML, OCR, `pdfJbIm`, `DML-CZ`, EuDML

Smaller is faster and safer too. (Stephen Adams, Google)

1 Motivation

Digital mathematical libraries (DMLs) contain a large volume of documents with scanned text (more than 80% of EuDML is scanned), which are mostly created by scanning older papers which were written and published earlier and their digital versions are already lost. Documents created this way are referred to as retro-born digital documents.

Research in math is influenced greatly by older articles and papers. When something new in math is discovered or researched, it is often based on older papers and discoveries. To make research more comfortable users require easy access to these kinds of documents. Thus DMLs need to provide documents that are easy to both find and access.

One user demand is for quick and easy access to documents. This means not only the ability to find where a document can be downloaded from, but also the ability to access it from the user's computer. The time to access a document is highly dependent on its size, which can be reduced using a good compression method. Documents in DMLs are mostly stored as PDF documents, which is probably the most widely-used document format on the Internet. In PDF, images are stored using various compression methods. One supported method is the JBIG2 compression method, which offers great compression ratios [1].

The bachelor thesis, JBIG2 Compression by Radim Hatlapatka [2], introduced a method for compressing PDF documents using the JBIG2 standard

with jbig2enc open-source encoder. This tool has been re-named pdfJbIm [3]. The improvement to jbig2enc introduced in this bachelor thesis improves the compression ratio of jbig2enc on average by a further 10%.

The results of the comparison of this tool with other tools have been published in [6]. In DML 2010, an article was published about the newer version of pdfJbIm and the results achieved with data stored in DML-CZ [7]. We are developing additional improvements of the jbig2enc encoder which use the results of an OCR engine to decide if two symbols are equivalent and thus should be stored only once in the dictionary. If they are found equivalent, the OCR engine can also help to decide which of them should be stored in the dictionary and be used for reconstructing an image when it is decompressed, thereby improving the quality of the image.

In this paper, we introduce the JBIG2 standard (see Section 2) and discuss issues that need to be addressed when OCR is used for compressing images to achieve the best possible results. We focus on issues connected to documents with math (see Section 3) and we describe a jbig2enc interface designed for using an OCR engine (see Section 4). Finally, we show some experimental results achieved when using Tesseract as the OCR engine (see Section 5).

2 Introduction to JBIG2

JBIG2 is a standard for compression of bitonal images developed by the Joint Bi-level Image Experts Group. These are images that consist of two colours only (usually black and white). The main area of such documents is a scanned text. JBIG2 was published in 2000 as an international standard ITU T.88 [9] and one year later as ISO/IEC 14492 [1]. It typically generates files that are three to five times smaller than Fax Group 4 and two to four times smaller than JBIG1, which was the previous standard released by the Joint Bi-level Image Experts Group [5]. JBIG2 also supports "perceptually lossless" coding. This is a special kind of lossy compression which causes no visually noticeable loss. Scanned text often contains flyspecks (tiny pieces of dirt) and perceptually lossless coding can help to get rid of the flyspecks and thus increase the quality of the output image.

The content of each page may be segmented into several regions with specific types of data. Most often it is segmented to a text region for text data, a halftone region for halftone images[1] and a generic region for the rest. In some situations, it is better to use the generic region for a specific type of data rather than a specific region such as the halftone region; in other situations, the converse is true.

The JBIG2 encoder segments text regions into components that most often correspond to symbols. For each set of equivalent symbols, one representant is chosen. The representant contains stored bitmap data and each occurrence of the symbol then directs to that representant with information about its position. These symbols must be encoded (both in the dictionary containing representants and also their occurrences). JBIG2 uses modified versions of Arithmetic and

[1] More about halftone can be found at `http://en.wikipedia.org/wiki/Halftone`

Huffman coding. Huffman coding is used mostly by faxes because of its lower computation demands, even though Arithmetic coding produces slightly better results.

JBIG2 supports a multi-page compression used for symbol coding (the coding of text regions). Any symbol that is used on more than one page is stored in a global dictionary. Such symbols need to be stored only once; space needed to store documents is thereby further reduced.

3 Specific Aspects of Using OCR in JBIG2 and Jbig2enc

OCR and image compression according to JBIG2 standard are very similar. In both cases, it is necessary to segment an image into components that are further processed. In OCR, there is necessary to detect text blocks and to detect individual symbols in order to be able to recognize them. In JBIG2, it is also necessary to detect text blocks and ideally also to detect individual symbols in order to be able to achieve the maximum compression ratio. There is one main difference between image compression that follows the JBIG2 standard, and OCR. In OCR, there is necessary to provide results even when OCR engine is uncertain, and to have the OCR engine trained to know the symbols contained in the image. When compressing an image with perceptually lossless compression encoder cannot afford errors but it has an advantage in that if it is uncertain about a symbol it can classify it as a new symbol, thereby preventing unwanted errors. It also does not need to know font information in advance.

When rendering text from images using OCR, the most important part is to recognize what is written, not in what format and fonts. This information is also welcome, but it is not the most important. If you want to compress an image there is necessary to differentiate between symbols in different fonts because the font information can be of value to the user. When using OCR in detecting equivalent symbols, it is necessary to take this information into account, and if this information is not provided by OCR itself, it needs to be handled by additional methods.

It is also necessary to take into account that atypical symbols can appear in documents, and they need to be handled correctly as well. From the OCR point of view, math symbols can be considered atypical. It is either possible to use a specialized OCR which handles math such as the Infty Reader [8], or to detect that it is math and process it specially.

4 Jbig2enc API for Using OCR

Our API consists of two parts (modules): one represents the data structure holding the results of OCR and one represents methods for running the OCR engine and retrieving its results.

Our goal is to make the API for holding OCR results and the API for using OCR engine as adaptable as possible in order to allow easy interchangeability of OCR engines and thus prevent unnecessary modifications of existing code.

Because jbig2enc is written in C++, our improvement and API also need to be in C/C++. We decided to use an object hierarchy which allows the creation of a common class with required methods; for creating new modules we used an inheritance. A new module using a different OCR engine is easily made by inheriting the relevant class and implementing defined methods. This methods are implemented specifically for the OCR engine that is actually used.

For holding OCR results, we need to allow the storage of additional data specific to the specific OCR engine that can be used to improve the comparison of representants and thus create a specific similarity function which is most suitable for that OCR engine.

Figure 1 shows classes representing the interface for using an OCR engine. On the left are classes representing the module for using an OCR engine and its function. Class `TesseractOcr` is an example of a module which uses Tesseract as the OCR engine. On the right classes holding results of OCR recognition are described. There is a main class for storing just simple structures with representative symbols. For holding OCR results, it is necessary to store additional information such as text recognized by the OCR engine and its confidence level. For this purpose, the class `OcrResult` is crated, which can be extended and thus new classed can easily be created to store additional information provided by the OCR engine.

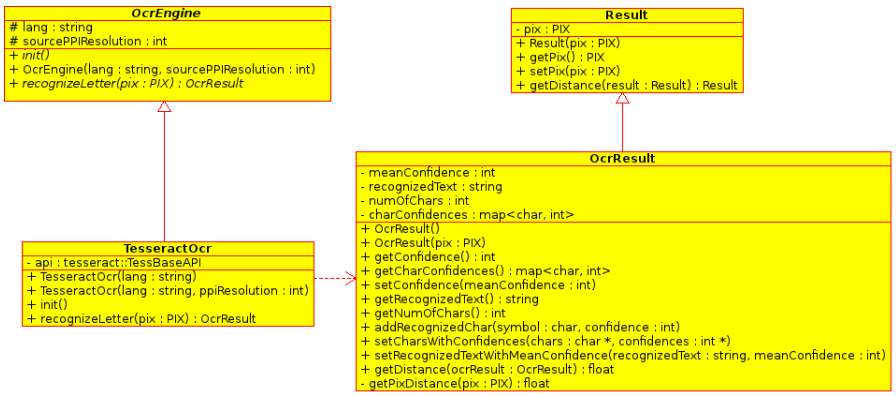


Fig. 1. Jbig2enc API for using an OCR engine

5 Experimental Results

In this Section, we introduce the results we achieved using our prototype version of the improved jbig2enc encoder which uses Tesseract as the OCR engine.

In order to show results of the created prototype, we compress set of more than 800 PDF documents. It is a set of PDFs selected randomly from collection

of Czech digital mathematical library. Documents are selected in order to cover different types of PDFs from different eras. For this purpose PDFs are chosen from different journals and from papers published in different years.

For compressing PDF documents pdfJbIm [3] is used. It uses our prototype version of the improved jbig2enc encoder. The prototype improves compression ratio by additional two percents in comparison with the previous improvement of the jbig2enc encoder [6].

The achievement is shown in Table 1 and in graph in Figure 2. All shown results are in KB. In order to prevent errors even on documents with an extremely bad quality, default thresholding value[2] is set to minimize potential loss of data. The documents also contain nonbitonal images that are ignored.

Table 1. Results of an enhanced jbig2enc encoder

Number of pages	Original PDF	Original jbig2enc	Improved jbig2enc without OCR	Improved jbig2enc with OCR
1	107.11	88.64 (82.8%)	86.72 (81%)	84.63 (79%)
2	240.76	203.19 (84.3%)	198.47 (82.4%)	193.83 (80.5%)
3	353.87	296.73 (83.9%)	288.11 (81.4%)	281.21 (79.5%)
4	476.82	401.13 (84.1%)	388.85 (81.6%)	379.38 (79.6%)
5	592.42	499.82 (84.4%)	484.31 (81.7%)	472.61 (79.8%)
6	722.71	609.02 (84.3%)	590.66 (81.7%)	576.42 (79.8%)
7	822.41	691.49 (84.1%)	667.13 (81.1%)	650.51 (79.1%)
8	949.18	800.55 (84.3%)	775.36 (81.7%)	756.16 (79.7%)
9	1,080.05	913.35 (84.6%)	880.55 (81.5%)	858.71 (79.5%)
10	1,161.09	975.56 (84%)	936.53 (80.6%)	913.19 (78.6%)

Figure 3 represents an original image (TIFF G4 compressed) that is further compressed according to the JBIG2 standard. The image size is 118 KB. Figures 4 (size 8.6 KB) and 5 (size 8.2 KB) are images compressed according JBIG2 standard using the jbig2enc open-source encoder [4]. Their sizes are around 8 KB which is a significant reduction from the original image. The difference between Figures 4 and 5 is that Figure 5 is compressed by the improved jbig2enc as described in Section 4. It uses Tesseract as the OCR engine.

In these figures, there is no visible loss of data or image quality, and without searching for differences in detail they look the same. Figure 6 shows the difference between the original image and the image compressed using the jbig2enc encoder which uses Tesseract as the OCR engine. Figure 7 shows how the output image changes when the jbig2enc encoder uses an OCR engine to improve its compression ratio, and as side effect has the potential to improve the quality of the output image.

[2] Thresholding value used by the jbig2enc encoder even if no improvement is used. It determines if two symbols should be considered equivalent or not

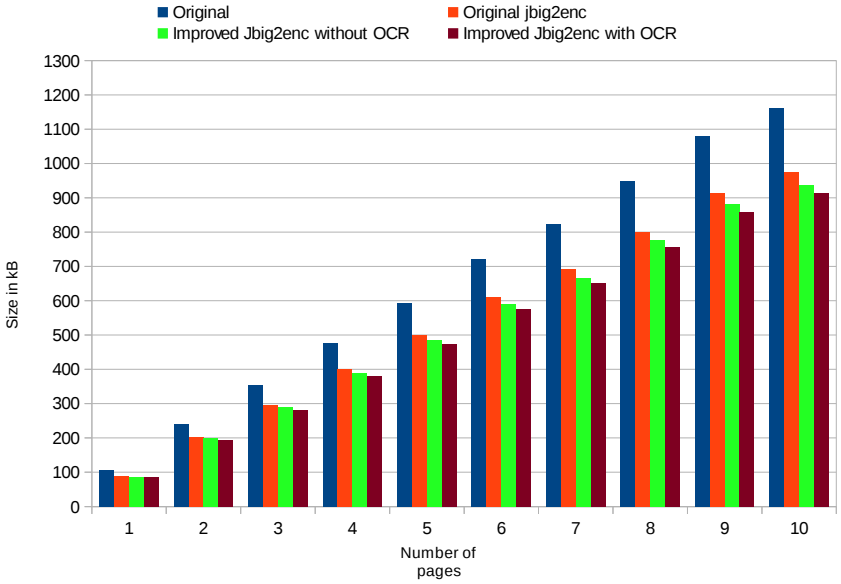


Fig. 2. Compression results of jbig2enc and its improved versions

6 Conclusion and Future Work

As we have shown, using OCR in jbig2enc further improves its compression ratio. We defined the API, which is independent of the OCR engine used. As a default OCR engine, we used the Tesseract OCR, but this can be easily replaced by another OCR engine by creating a new module specifically for use with that OCR engine which implements the defined API and by changing one line in the existing code to determine which OCR engine is used.

We have shown that by using OCR, we are able to choose which representant of several equivalent ones is better in terms of visual quality, thus improving the quality of the image.

The similarity distance function is not yet fully balanced for simultaneously maximizing the compression ratio and preventing errors. This needs to be solved and well tested. For this purpose, we intend to create semi-automatic testing by experimenting on images for which we know output will be created correctly without quality reduction, and for which we know a recognized number of different symbols. We shall then simulate the decrease in quality caused by scanning the image and see to what extent the result changes.

We intend to create a universal language dictionary containing all of the symbols used in European languages, including math symbols, and train Tesseract for it. This should prevent the need for the user to set all used languages and improve speed if more languages are used. More dictionaries means recurrence of the same symbols, thus creating a bigger collection than necessary. While

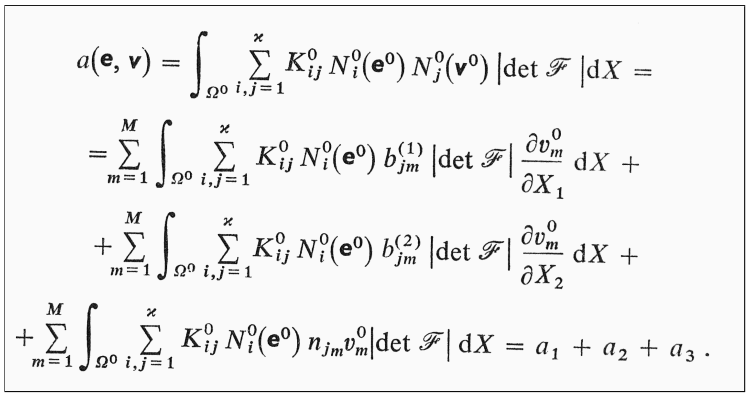


Fig. 3. Original image before JBIG2 compression (TIFF G4 compressed, size: 118 KB)

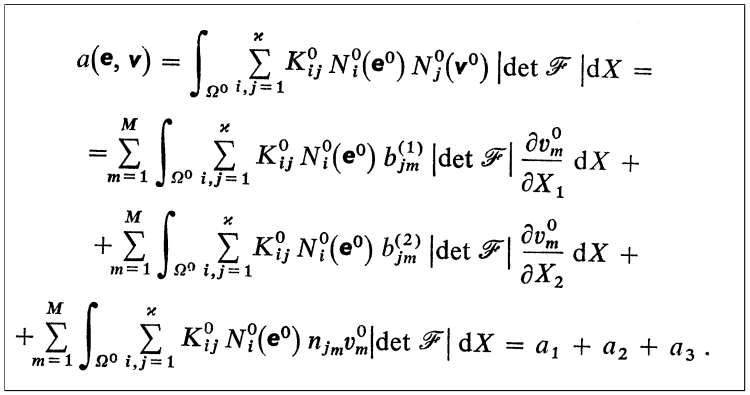


Fig. 4. Images compressed according to standard JBIG2 without OCR usage (size: 8.6 KB)

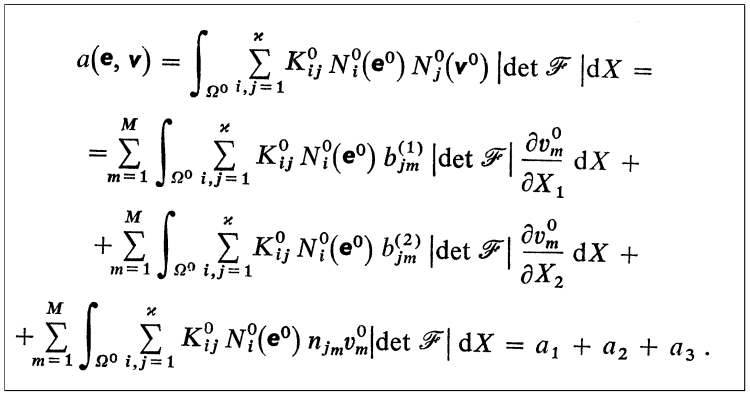


Fig. 5. Image compressed according to standard JBIG2 with OCR usage (size: 8.2 KB)

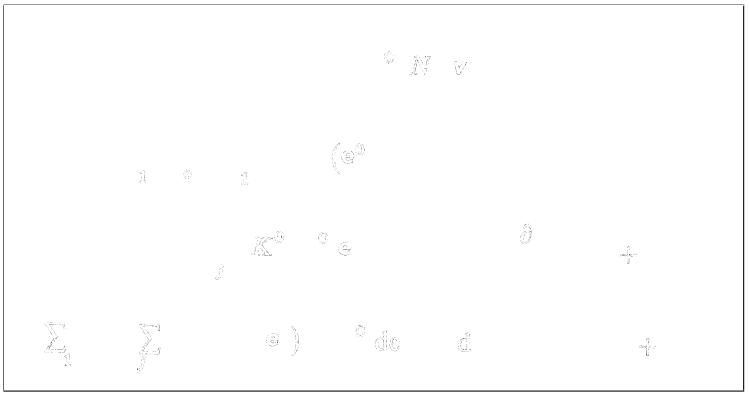


Fig. 6. Difference between Fig. 3 and Fig. 5

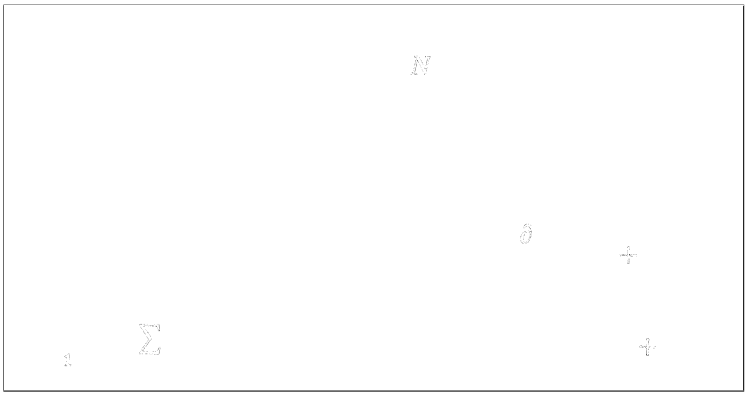


Fig. 7. Difference between Fig. 4 and Fig. 5

recognizing symbols, the OCR engine compares data with the existing collection of symbols created based on the dictionaries provided to determine what symbol is represented by the image. The size of this collection influences the amount of comparisons needed to determine which is the best candidate.

There is also a plan for creating a module for using Infty as the OCR engine and to use its math recognition support to improve the compression ratio for documents containing lots of math.

Acknowledgement

This work has been financed in part by the European Union through its Competitiveness and Innovation Programme (Information and Communication Technologies Policy Support Programme, "Open access to scientific information", Grant Agreement No. 250503).

References

1. Committee, J.: 14492 FCD. ISO/IEC JTC 1/SC 29/WG 1 (1999), http://www.jpeg.org/public/fcd14492.pdf
2. Hatlapatka, R.: JBIG2 komprese (Bachelor thesis written in Czech, JBIG2 compression). Masaryk University, Faculty of Informatics (advisor Petr Sojka), Brno, Czech Republic (2010)
3. Hatlapatka, R.: PDF Recompression using JBIG2. [online] (2012), http://nlp.fi.muni.cz/projekty/eudml/pdfRecompression/
4. Langley, A.: Homepage of jbig2enc encoder. [online], http://github.com/agl/jbig2enc
5. SG, S.J.: JBIG Maui Meeting Press Release (December 1999), http://www.jpeg.org/public/mauijbig.pdf
6. Sojka, P., Hatlapatka, R.: Document engineering for a digital library: PDF recompression using JBIG2 and other optimizations of PDF documents. In: Proceedings of the 10th ACM symposium on Document engineering. pp. 3–12. DocEng '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1860559.1860563
7. Sojka, P., Hatlapatka, R.: PDF Enhancements Tools for a Digital Library: pdfJbIm and pdfsign. In: DML 2010 Towards a Digital Mathematics Library. pp. 45–55. Masaryk University, Brno, Czech Republic (2010)
8. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFTY – An Integrated OCR System for Mathematical Documents. In: Proceedings of ACM Symposium on Document Engineering 2003. ACM, Grenoble (2003), http://www.inftyproject.org/articles/2003_DocEng_Suzuki.zip
9. Union, I.T.: ITU-T Recommendation T.88. ITU-T Recommendation T.88 (2000), http://www.itu.int/rec/T-REC-T.88-200002-I/en

Normalization of Digital Mathematics Library Content

MathML Canonicalization

David Formánek, Martin Líška, Michal Růžička, and Petr Sojka

Masaryk University, Faculty of Informatics
Botanická 68a, 602 00 Brno, Czech Republic
david.formanek@mail.muni.cz, martin.liski@mail.muni.cz,
mruzicka@mail.muni.cz, sojka@fi.muni.cz

Abstract. Paper discusses the needs for data normalization in a Digital Mathematics Library (DML). Specifically, emphasis is given to canonicalizing formulae encoded in Presentation MathML notation which starts to be available in several DMLs and is used by DML applications. This is a prerequisite for advanced processing — namely math enabled fulltext searching or semantic filtering and automated classification. Different sources of MathML and their specifics are described. Several use cases of possible formulae canonicalization transformations are listed and discussed in detail. Findings are finally concluded and a design of a to-be-developed canonicalization tool is outlined.

Keywords: MathML normalization, canonicalization, digital mathematics libraries, DML, presentation MathML

1 Motivation

Modern Digital Mathematics Libraries (DML) such as EuDML [18,5] base their services on paper semantics, i.e. fulltext handling, including mathematical formulae, as well as basic metadata and Mathematics Subject Classification (MSC) codes. Mathematics literature is widely dispersed across a high number of publishers, making it very difficult to collect fulltexts from these heterogeneous sources. This situation is very different from other libraries, such as PubMed Central for biomedical and life sciences, where publishers have an agreed workflow using the NLM Journal Publishing Tag Set and tools developed with funding from the National Institutes of Health.

Full paper texts have to be 'homogenized', converted to some uniform representation, in order for math-aware full-text searches [15] and paper similarity computations [11,12] to work properly. These tasks are usually handled based on a bag-of-words representation of a document text — vector space model — every term (word, lemma) has its own dimension and the number of occurrences of a term reflects its value. Non-textual terms such as mathematical formulae are mostly not taken into account. This creates another challenge for DMLs, as

mathematical formulae are the essence of mathematical publications. There is an average of 380 mathematical formulae per arXiv paper in the MREC database [8]. It has been reported [21] that even a single histogram of mathematical symbols is sufficient for domain classification of a paper in the mathematical domain.

To reliably represent a paper for DML processing, including handling the mathematics, it is necessary to

1. select a canonical representation of the non-textual *structural* entities appearing in fulltexts (mathematical symbols, formulae, and equations); and
2. decide on equivalence classes for these entities (e.g., for which formulae should be considered equal for given DML tasks such as search, similarity computation, formulae editing, and conversion of math into Braille).

In this paper, we discuss the options for selecting the canonical representations of formulae to be used in DML tools, and the *canonicalization* process — the process — of computing this canonical representation from a variety of different sources and formats.

Our primary motivation is the natural requirement for our own (Web)MIaS system, which currently uses Presentation MathML [14] to operate correctly and offer an expected search behaviour to users regardless of the MathML input source. When a user posts a query to the system, the system must abstract it from the underlying notational differences in order for it to behave correctly. This requirement is increasingly emphasized with the growing number of different sources of MathML. Currently there are three sources (LaTeXML, Tralics, and user input; the number is expected to increase). If they are not correctly normalized the system misbehaves and it appears to users as if it simply does not work, however good the underlying design is.

We have used UMCL library [1,2] for canonicalization in our MIaS system sofar. However, we have found that the deficiencies of the software are so severe (change of formulae semantics, slowness,...) [7, chapter 5], and the need for canonicalization so important, that we have decided to design and implement new canonicalization tool from scratch.

This paper is structured as follows: in Section 2, different sources of mathematics are described and their differences are discussed. The core part of this paper is Section 3, where several use cases of possible canonical representation and canonicalization are documented and suggested. We conclude with Section 5, and present a plan for future work.

2 MathML Sources

To store mathematical formulae in our documents we have chosen MathML[1] — an XML-based language — as a widely used, formally defined, but still evolving standard. The widespread use of MathML and its XML base means of this

[1] More precisely, Presentation MathML, as there are currently significantly more real-life resources using this form of MathML than Content MathML.

language is supported by various tools in the whole document workflow. More importantly, MathML can be used as a common language among the advanced computer mathematical software packages that are extensively used by working mathematicians.

On the author end of the document workflow the MathML code can be 'hand made' using simple plain text editors such as MS Windows Notepad, or something more comfortable, such as specialized XML editors that are usually part of various integrated development environments. For example, the formula $x^2 + y^2$ can be written as follows:

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
    <msup>
        <mi>x</mi><mn>2</mn>
    </msup>
    <mo>+</mo>
    <msup>
        <mi>y</mi><mn>2</mn>
    </msup>
</math>
```

Listing 1: Example of the 'hand made' formula $x^2 + y^2$

However, the XML nature of MathML makes the coding of more complex formulae rather long for manual construction. Various software tools are more frequent sources of MathML. MathML can be generated as an output / data exchange format of complex specialized programs, such as Maple, Matlab, and Mathematica [9,20,22], or web services, such as the well known Wolfram Alpha [23], that are extensively used by mathematicians to support their work.

```
generate::MathML(x^2 + y^2,
                 Content = FALSE, Annotation = FALSE)
<math xmlns='http://www.w3.org/1998/Math/MathML'>
  <mrow xref='No7'>
    <msup xref='No3'>
      <mi xref='No1'>x</mi>
      <mn xref='No2'>2</mn>
    </msup>
    <mo>+</mo>
    <msup xref='No6'>
      <mi xref='No4'>y</mi>
      <mn xref='No5'>2</mn>
    </msup>
  </mrow>
</math>
```

Listing 2: Example of MathML export of the formula $x^2 + y^2$ by Matlab 7.9.0 MuPAD symbolic engine

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
 <mrow>
  <msup>
   <mi>x</mi>
   <mn>2</mn>
  </msup>
  <mo>+</mo>
  <msup>
   <mi>y</mi>
   <mn>2</mn>
  </msup>
 </mrow>
</math>
```

Listing 3: Example of the MathML export of the Wolfram Alpha input query 'x^2 + y^2'

On the consumer end of the document workflow MathML can be used as an input for mathematical programs and services (Maple, Matlab, Mathematica, Wolfram Alpha, etc.) or simply displayed — usually as part of an XHTML web page — in a web browser with MathML support.

However, a large number of mathematical documents are produced using the TEX typesetting system and authored in TEX markup. Thus, it is necessary to be able to convert the TEX source code of mathematical formulae to the MathML language. Our main motivation is the WebMIaS system. For more complex input formulae, it would be uncomfortable for the user to manually construct queries in MathML, as the code would be very complicated. The well known LATEX syntax is far more appropriate for manual input. Therefore, we need a conversion from LATEX to MathML as part of the WebMIaS input routine.

There are several tools that are able to convert TEX markup to the MathML language. For example, arXMLiv [16] employs LATEXML [19]. The EuDML project and our WebMIaS [8] system internally use Tralics [6].

```
<math xmlns="http://www.w3.org/1998/Math/MathML"
    alttext="x^{2}+y^{2}" display="inline">
    <semantics>
        <mrow>
            <msup><mi>x</mi><mn>2</mn></msup>
            <mo>+</mo>
            <msup><mi>y</mi><mn>2</mn></msup>
        </mrow>
        <annotation encoding="application/x-tex">
            x^{2}+y^{2}
        </annotation>
    </semantics>
</math>
```

Listing 4: Example of LATEXML generated MathML of formula $x^2 + y^2$

```
<math xmlns='http://www.w3.org/1998/Math/MathML'>
    <mrow>
        <msup>
            <mi>x</mi> <mn>2</mn>
        </msup>
        <mo>+</mo>
        <msup>
            <mi>y</mi> <mn>2</mn>
        </msup>
    </mrow>
</math>
```

Listing 5: Example of Tralics generated MathML of formula $x^2 + y^2$

A frequent type of mathematical document in DML is the older papers that are unavailable in any digital-format or are available only in an 'end' format such as PDF that is suitable for reading and printing but is not appropriate for direct MathML processing. These documents can be a significant part of the DML content collection, so they are worth further processing.

Documents available in hard copy only can be scanned and processed using InftyReader [17] optical character recognition (OCR) software. InftyReader has a unique feature for detecting mathematical formulae in a scanned document. These formulae can be subsequently saved as MathML.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
    <msup>
        <mi mathvariant="italic">x</mi>
        <mrow>
            <mn mathvariant="normal">2</mn>
        </mrow>
    </msup>
    <mo mathvariant="normal">+</mo>
    <msup>
        <mi mathvariant="italic">y</mi>
        <mrow>
            <mn mathvariant="normal">2</mn>
        </mrow>
    </msup>
</math>
```

Listing 6: Example of InftyReader generated MathML from a PDF document containing only formula the $x^2 + y^2$ in its body

Born-digital PDF documents with no available source codes can be processed using the MaxTract software [3,4], which that is under intensive development as part of the EuDML project. MaxTract generates LaTeX source / XHTML+MathML representation of the document based on an optical analysis of the positions of

characters on the page. The analysis is supported with information from the fonts embedded in the processed document.

```
<math display="block" xmlns="&mathml;">
 <mi>&#x0078;</mi>
 </math>

<p >

</p>

<p align="right" >
<math display="inline" xmlns="&mathml;">
 <mi>&#x0079;</mi>
 </math>
</p>

<p >

</p>
```

Listing 7: Example of XHTML + MathML generated by the development version of MaxTract from a PDF document containing only the formula $x^2 + y^2$ in its body

During the MathDex project, it became clear that the most time- and resources-consuming task in building a math search engine and database is the normalization and conversion of heterogeneous sources [10]. As shown in Listings 1 — 6, MathML can vary slightly due to the different ways a code was obtained, even for a trivial formula like $x^2 + y^2$.

In a DML project, there can be differences in the final MathML encoding even for semantically and structurally similar formulae, due to the origins of the MathML from different sources. In Section 3, several more complicated examples of possible ambiguities in MathML are discussed that have to be normalized to allow math searches and similarity computation.

3 Use Cases

Using our public working demo of the WebMIaS system we discovered several discrepancies in the form of MathML generated by the real-time TEX to MathML converter we currently use — Tralics — and by the MathML canonicalizer from the UMCL library. We employed the UMCL canonicalization module to try to normalize the users' MathML input and the MathML produced by the LATEXML converter contained in the arXMLiv collection. Then we went through the Presentation MathML specifications and gathered a list of possible reformatting rules we could perform.

The goal is to reduce the possible MathML scripts with the same semantics and mathematical structures to just one representation. To have such a canonicalized representation is convenient for many applications, as was described in Sections 1 and 2.

Analyzing the issues of possible inconsistencies and ambiguities of MathML-encoded formulae raised design and strategy questions. Conceptual decisions for handling different types of similar constructions and completely different formulae need to be made.

More specifically, for example, should we try to keep the MathML compact and reduce the number of nodes in transformations, or should we try to add nodes for better disambiguation? Another question is: should our future canonicalization tool produce valid MathML according to this schema? Unquestionably, this feature would be nice to have for many reasons and possible applications, but it certainly adds more requirements and takes much more effort to design and implement not only true/false validation, but also functional correctness validation.

Below are described proposals and discussions of transformations that can be performed with relatively minor difficulty. The list is not complete and is subject to further evaluation.

3.1 Removing Elements and Attributes

Many of the MathML elements used in Presentation MathML make little or no contribution to the semantics of the formula and therefore also to the formulae for indexing and searching. These are usually elements that alter the appearance of formulae in some way — space-like elements such as `mspace`, `mpadded`, `mphantom`, `maligngroup`, and `malignmark`. They may occasionally have some semantic meaning, but we prefer to canonicalize similar formulae into one representation rather than risk treating the same formulae as different. Therefore, these elements are best omitted. The content of the `mtext` element should be indexed as normal text before removal.

Most element attributes are similarly undesirable. Many are used for formatting, affecting only the appearance of rendered formulae (for example, the attributes `linebreak` and `indentalign` of the `mo` element). Others might have some slight semantic significance, but are very uncommon and usually not very important; we think these attributes should be removed. However, several exceptions exist. For instance, the element `mfrac` is used for fractions but its meaning changes with the attribute `linethickness` set to 0, which express a binomial coefficient. The attributes of the element `mfenced` are also important (see Listing 9). The attribute `mathvariant` can also influence formula semantics and therefore should be preserved in all possible elements. For example, the MIaS system makes use of this attribute so that hits with the assigned `mathvariant` font specifying the attribute are more relevant.

```
<mfrac>
  <mrow>
    <mi> x </mi>
    <mo> + </mo>
    <mi> y </mi>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
  <mrow>
    <mi> x </mi>
    <mphantom>
      <mo> + </mo>
      <mi> y </mi>
    </mphantom>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
</mfrac>
```

```
<mfrac>
  <mrow>
    <mi> x </mi>
    <mo> + </mo>
    <mi> y </mi>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
  <mrow>
    <mi> x </mi>
    <mo> + </mo>
    <mi> z </mi>
  </mrow>
</mfrac>
```

Listing 8: Example of `<mphantom>` ommision

```
<mfrac linethickness="2"
       bevelled="true">
  <mi> a </mi>
  <mi> b </mi>
</mfrac>
```

```
<mfrac>
  <mi> a </mi>
  <mi> b </mi>
</mfrac>
```

Listing 9: Example of omission of unnecessary attributes in `mfrac`

3.2 Unifying Fences

There are two approaches to creating fenced formulae. One is more semantic and uses the `mfenced` element with the `open`, `close`, and `separator` attributes to describe delimiters and separators. The other places fence symbols directly within `mo` elements, and the fenced formula is enclosed in the `mrow` element to group the elements together. Although the first approach seems to be valid, we prefer the second one as it is more universal and allows easier conversion — e.g., converting addition to `mfenced` with attribute `separators` set to `+` would be invalid. As shown in Listing 10, `mfenced` elements are replaced by a more general `mrow` element, and fence and separator symbols are added as `mo` elements. Fenced elements are further enclosed in an `mrow` element so it can be treated as a single expression when needed. We could also consider unifying the symbols used as separators/delimiters.

```
<mfenced open="[">

  <mi> x </mi>

  <mi> y </mi>

</mfenced>
```

```
<mrow>
  <mo> [ </mo>
  <mrow>
    <mi> x </mi>
    <mo> , </mo>
    <mi> y </mi>
  <mrow>
  <mo> ) </mo>
</mrow>
```

Listing 10: Two ways of writing interval $[x, y)$

3.3 Mrow Minimizing

The `mrow` element is used for grouping other elements. Its most common use case is to obtain a given correct number of child elements of some parent element (e.g. `mfrac` needs two child elements). We can determine unnecessary occurrences of `mrow` by summing the number of its child elements and its siblings with respect to the number of required elements for the parent element. Parents requiring only one child element actually accept any number of elements that are treated as if they are inferred within a single `mrow` element. Hence, the grouping element is redundant and can be removed. In any case, the impact of the transformations to any form of processing canonicalized notation must be taken into account and the structure of the formulae cannot be violated. For instance, after removing the `mfenced` enclosing element we ought to wrap the fenced formula with an `mrow` if it is not.

```
<msqrt>
  <mrow>
    <mo> - </mo>
    <mn> 1 </mn>
  </mrow>
</msqrt>
```

```
<msqrt>

  <mo> - </mo>
  <mn> 1 </mn>

</msqrt>
```

Listing 11: Example of `<mrow>` removal after optimization $\sqrt{-1}$

3.4 Sub-/Superscripts Handling

The `msubsup` element used for attaching subscript and superscript to another element at the same time is redundant — the same thing can be expressed as

a combination of `msub` and `msup` elements. The order of the elements is important. When both elements are used, we prefer to place `msub` within `msup` (see Listing 12) because a subscript is usually more closely related to the base expression. A similar problem and solution is related to the elements triad of `munder`, `mover`, and `munderover`. Both `msubsup` and `munderover` can be used for limits of integration or bounds of summations; therefore, we should use only one canonical representant.

```
<msubsup>
  <mi> x </mi>

  <mn> 1 </mn>

  <mn> 2 </mn>
</msubsup>
```

```
<msup>
  <msub>
    <mi> x </mi>
    <mn> 1 </mn>
  </msub>
  <mn> 2 </mn>
</msup>
```

Listing 12: Two ways of expressing x_1^2

3.5 Applying Functions

There are many ways to express functions. Entity `&#x2061;` (function application) should be used but we cannot rely on that, so we suggest removing this operator for the purpose of unification. The opposite approach — adding the function application operator where it was omitted — could be rather tricky and could lead to ambiguities. The name of the function should occur in the `mi` element but it also can be considered as an operator and be placed in the `mo` element. The arguments of a function can be fenced with parentheses or an `mfenced` element or both. We chose canonical representation without an entity, with `mrow` and parentheses (see Listing 14). Other ambiguities can be caused by different invisible operators. For example, two identifiers in a subscript with no operator usually means multiplication but it can mean separation too.

4 Design Considerations

The design and implementation decisions of the canonicalization application depend on the purpose of new canonicalizer. Even though the use of the math content by different tools might be similar, the experience shows that we hardly could 'fit one size' for all applications. Thus the main design imperative is the modularity, simplicity, extensibility and flexibility, so that the canonicalizer might be easily modified when the need of the applications change. With different data the canonicalizer might change even for different types of math-aware search.

```
<mi> f </mi>
<mo> &#x2061; </mo>
<mrow>
  <mo> ( </mo>
  <mi> x </mi>
  <mo> ) </mo>
</mrow>
```

```
<mi> f </mi>
<mrow>
  <mo> ( </mo>
  <mi> x </mi>
  <mo> ) </mo>
</mrow>
```

Listing 13: Using or not using the operator for function application

```
<mi> sin </mi>

<mo> &#x2061; </mo>

<mi> x </mi>
```

```
<mi>sin</mi>
<mrow>
  <mo>(</mo>
  <mi>x</mi>
  <mo>)</mo>
</mrow>
```

Listing 14: Adding parentheses to sine function argument

Examples in subsections of previous section form set of modules that do the necessary MathML tree transformations as recursive procedures on MathML trees.

According to the expected size of the input data set, effectiveness, the speed of the canonicalization application is also a critical parameter — in our MREC [8] corpora there is 168,000,000 formulae to canonicalize. Thus, use of standard XSL transformations does not seem to be appropriate, for example, as UMCL example showed.

Another key decision is handling of invalid input MathML and question of valid MathML on the output as mentioned in Section 3.

As the (Web)MIaS system as well as other core parts of EuDML system (Lucene) do use the Java platform is seems to be natural to use Java also for the implementation of canonicalization application.

5 Conclusions and Future Work

We consider MathML canonicalization important for proper functioning of several math-aware applications that handle documents in DMLs. We have defined the problems and enumerated the most important use cases as modules of newly designed canonicalizer.

We are currently working on finishing the design and implementation of a first version of application that will be used for the task of math indexing in MIaS

system employed in EuDML project. By evaluation of this task we will verify our design decisions and plan to use it for another tools working with math fulltext data (semantic similarity tools as gensim [12]).

Acknowledgements This work was partially supported by the European Union through its Competitiveness and Innovation Programme (Information and Communication Technologies Policy Support Programme, 'Open access to scientific information', Grant Agreement No. 250503, a project of the European Digital Mathematics Library, EuDML).

References

1. Archambault, D., Berger, F., Moço, V.: Overview of the "Universal Maths Conversion Library". In: Pruski, A., Knops, H. (eds.) Assistive Technology: From Virtuality to Reality: Proceedings of 8th European Conference for the Advancement of Assistive Technology in Europe AAATE 2005, Lille, France. pp. 256–260. IOS Press, Amsterdam, The Netherlands (Sep 2005)
2. Archambault, D., Moço, V.: Canonical MathML to Simplify Conversion of MathML to Braille Mathematical Notations. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (eds.) Computers Helping People with Special Needs, Lecture Notes in Computer Science, vol. 4061, pp. 1191–1198. Springer Berlin / Heidelberg (2006), http://dx.doi.org/10.1007/11788713_172
3. Baker, J.B., Sexton, A.P., Sorge, V.: A linear grammar approach to mathematical formula recognition from PDF. In: Proceedings of the Conferences in Intelligent Computer Mathematics, CICM 2009. LNAI, vol. 5625, pp. 201–216. Springer (2009)
4. Baker, J.B., Sexton, A.P., Sorge, V.: Towards reverse engineering of PDF documents. In: Sojka, P., Bouche, T. (eds.) Towards a Digital Mathematics Library, DML 2011. pp. 65–75. Masaryk University Press, Bertinoro, Italy (July 2011), http://hdl.handle.net/10338.dmlcz/702603
5. Borbinha, J., Bouche, T., Nowiński, A., Sojka, P.: Project EuDML—A First Year Demonstration. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) Intelligent Computer Mathematics. Proceedings of 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011. Lecture Notes in Artificial Intelligence, LNAI, vol. 6824, pp. 281–284. Springer-Verlag, Berlin, Germany (Jul 2011), http://dx.doi.org/10.1007/978-3-642-22673-1_21
6. Grimm, J.: Producing MathML with Tralics. In: Sojka [13], pp. 105–117, http://dml.cz/dmlcz/702579
7. Jarmar, M.: Conversion of Mathematical Documents into Braille. Master's thesis, Faculty of Informatics (Jan 2012), https://is.muni.cz/th/172981/fi_m/?lang=en
8. Líška, M., Sojka, P., Růžička, M., Mravec, P.: Web Interface and Collection for Mathematical Retrieval. In: Sojka, P., Bouche, T. (eds.) Proceedings of DML 2011. pp. 77–84. Masaryk University, Bertinoro, Italy (Jul 2011), http://www.fi.muni.cz/~sojka/dml-2011-program.html
9. Maplesoft, a division of Waterloo Maple Inc.: MathML – Maple Help (Apr 2012), http://www.maplesoft.com/support/help/Maple/view.aspx?path=MathML
10. Munavalli, R., Miner, R.: MathFind: A Math-Aware Search Engine. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 735–735. SIGIR '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1148170.1148348

11. Řehůřek, R., Sojka, P.: Automated Classification and Categorization of Mathematical Knowledge. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) Intelligent Computer Mathematics—Proceedings of 7th International Conference on Mathematical Knowledge Management MKM 2008. Lecture Notes in Computer Science LNCS/LNAI, vol. 5144, pp. 543–557. Springer-Verlag, Berlin, Heidelberg (Jul 2008)
12. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en, software available at http://nlp.fi.muni.cz/projekty/gensim
13. Sojka, P. (ed.): Towards a Digital Mathematics Library. Masaryk University, Paris, France (Jul 2010), http://www.fi.muni.cz/~sojka/dml-2010-program.html
14. Sojka, P., Líška, M.: Indexing and Searching Mathematics in Digital Libraries (Mar 2011), submitted to MKM 2011
15. Sojka, P., Líška, M.: Indexing and Searching Mathematics in Digital Libraries – Architecture, Design and Scalability Issues. In: Davenport, J.H., Farmer, W.M., Urban, J., Rabe, F. (eds.) Intelligent Computer Mathematics. Proceedings of 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011. Lecture Notes in Artificial Intelligence, LNAI, vol. 6824, pp. 228–243. Springer-Verlag, Berlin, Germany (Jul 2011), http://dx.doi.org/10.1007/978-3-642-22673-1_16
16. Stamerjohanns, H., Kohlhase, M., Ginev, D., David, C., Miller, B.: Transforming Large Collections of Scientific Publications to XML. Mathematics in Computer Science 3, 299–307 (2010), http://dx.doi.org/10.1007/s11786-010-0024-7
17. Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., Kanahori, T.: INFTY — An integrated OCR system for mathematical documents. In: Vanoirbeek, C., Roisin, C., Munson, E. (eds.) Proceedings of ACM Symposium on Document Engineering 2003. pp. 95–104. ACM, Grenoble, France (2003)
18. Sylwestrzak, W., Borbinha, J., Bouche, T., Nowiński, A., Sojka, P.: EuDML—Towards the European Digital Mathematics Library. In: Sojka [13], pp. 11–24, http://dml.cz/dmlcz/702569
19. The LaTeXML project: The LaTeXML Developer Portal (Apr 2012), https://trac.mathweb.org/LaTeXML/
20. The MathWorks, Inc.: MuPAD – Matlab (May 2012), http://www.mathworks.com/discovery/mupad.html
21. Watt, S.M.: Mathematical Document Classification via Symbol Frequency Analysis. In: Sojka, P. (ed.) Towards Digital Mathematics Library—Proceedings of DML 2008. pp. 29–40. Masaryk University, Birmingham, UK (Jul 2008), http://www.fi.muni.cz/~sojka/dml-2008-program.xhtml
22. Wolfram: Mathematica Import/Export Format : MathML (Apr 2012), http://reference.wolfram.com/mathematica/ref/format/MathML.html
23. Wolfram Alpha LLC: Wolfram Alpha (Apr 2012), http://www.wolframalpha.com/