

Knowledge Representation and Management in ACTIVEMATH

The ACTIVEMATH group (dev@activemath.org):

Erica Melis, Jochen Büdenbender, George Goguadze, Paul Libbrecht
and Carsten Ullrich

DFKI Saarbrücken, Germany

Abstract. ACTIVEMATH is an open web-based learning environment for mathematics. It dynamically generates interactive mathematical courses adapted to the student's goals, preferences, capabilities, and knowledge. Its content is represented in an extended OMDoc which in turn is an extension of the OpenMath XML-language. ACTIVEMATH is the first system that uses OMDoc. It makes use of this knowledge representation in several knowledge management tasks, among them the Web-presentation of mathematical text including formulae, the communication with the integrated mathematical systems, the user-adaptive composition of a course, the dynamic generation of learning suggestions, and the dictionary browsing.

The bias of the knowledge representation and management towards mathematics is obvious in ACTIVEMATH because it meets the challenge of the presentation of mathematical formulae on the Web together with definitions, proofs, and exercises using mathematical systems.

Keywords: representation of mathematical knowledge, web-based, adaptive learning environment, Web publishing of mathematics, interactivity, adaptivity, semantics, metadata

1. Introduction

ACTIVEMATH¹ is an open web-based learning environment that dynamically presents interactive mathematical courses adapted to the student's goals, preferences, capabilities, and knowledge. Its knowledge representation is reusable in other contexts. It integrates several mathematical systems for explorative learning and interactive exercises. Given these characteristics, what kind of knowledge representation does such a system need?

First of all, the knowledge representation has to be *separated* from the functionalities of the system: this ensures that the adaptivity, interactivity, and other features of the system are truly extensible and modifiable. This separation is also required for the reusability and interoperability of the encoded content.

Secondly, the knowledge representation has to provide a content *structure* with conceptual units appropriate for *mathematical* learning documents.

Third, it has to be *annotated* with information that supports reasonable search facilities, user-adativity in choosing the content as well as coherently and appropriately presenting it and reacting to the user's actions.

Fourth, it needs to comprise the definition and semantics of mathematical *objects* in order to guarantee machine-readability. This is a basis for the inter-operability of external (mathematical) systems integrated into an educational system. Such a representation of actual mathematical objects is also a condition for the *re-usability* of the content in different contexts and for a *semantic handling* of formulae.

This article shows how ACTIVEMATH's knowledge representation satisfies the four requirements and how it employs the knowledge representation for its functionalities currently and in the near future. In addition, we compare our knowledge representation to XHTML+MATHML and L^AT_EX which are the current standards in mathematical publishing on the Web.

2. Knowledge Representation

Many on-line educational documents have been produced in recent years, hence the obvious objective of *reusing* the once encoded content arises. Another naturally occurring objective is the *inter-operability* of systems that function as expert services for mathematical work and education – such as Computer Algebra Systems (CASs), theorem provers, search engines, or course generators.

For the re-usability of material and the machine readability of data, ontologies, i.e. a definition of the conceptual units of a domain and their relationships, can be used. If two systems use a common ontology, knowledge sharing can be supported. If they use different ontologies for the same domain, it is necessary to translate from one ontology to the other.

2.1. KNOWLEDGE REPRESENTATION IN THE WEB

The construction of common ontologies can be seen as a first step to introduce semantics, in particular in knowledge sources on the Web. Therefore standardized ontologies and the technologies to encode and decode content that is represented in these ontologies are the backbone for the vision of the semantic Web² which understands the World Wide Web as an enormous and fast growing collection of knowledge sources that can be shared and processed by automated tools as well as by humans.

Currently, mathematical knowledge sources use one of the following categories of representation:

- a purely formal representation
- a structural and/or semantic markup
- a purely presentation-oriented markup

Each of these representations have advantages. Formal content, such as the automated provers' libraries (e.g. the TPTP language or the HEAM library), can be handled by machines but is difficult to present in a readable format.

Content represented using presentational markup, such as \TeX or xHTML+MATHML -presentation is easily authored and elaborate layout tools exist. It is not machine-readable and does not carry semantics through.

The current framework that provides the opportunity to structure content and to identify particular classes of content components is the eXtensible Markup Language XML. In this framework, the structure of any class of documents can be encoded. The structure is specified in a Document Type Definition (DTD) which defines a grammar (an "XML language") for the logical structure of a document class. The DTD can be extended.

Semantically represented content contains units with texts and semantically encoded formulae, it can be presented with reasonable effort, it can be managed efficiently, and its semantic parts are machine-readable. This article describes the semantic knowledge representation and its management that is at the heart of the ACTIVE MATH server. Moreover, it describes its re-usability and presentation features.

2.2. KNOWLEDGE REPRESENTATION IN ACTIVE MATH

For the ACTIVE MATH system, the reuse of content in different contexts is particularly important because its user-adaptivity requires that the same content can be presented in different ways depending on the user and on the learning situation. Machine-readability of the knowledge is needed, as ACTIVE MATH integrates external systems such as the proof planner Ω MEGA [11] and the CAS MUPAD [14].

2.2.1. OMDoc

ACTIVE MATH has been the first system whose knowledge representation is OMDoc [10]. OMDoc is an extension of the OpenMath XML-standard.³ OpenMath provides a grammar for the representation of

mathematical objects and sets of standardized symbols (the content-dictionaries). There exist phrase-books for the conversion of **OpenMath** expressions to the syntax of different CASs, among them Maple [3], Mathematica [15], and GAP [13].

The **OpenMath** content dictionaries cover only a few mathematical disciplines and in general could never be complete since mathematicians tend to frequently define new symbols. Apart from the incomplete content dictionaries, **OpenMath** does not provide any means to structure the content of a mathematical document by dividing it into its logical units such as “definition”, “theorem”, and “proof”.

Therefore, **OMDoc** was developed by Michael Kohlhase. It inherits the grammar for mathematical objects from **OpenMath** and the existing content dictionaries. In addition, **OMDoc** defines a framework for the definition of new symbols. This allows content authors to augment the incomplete **OpenMath** content dictionaries. New symbols can be introduced where they are needed and definitions can be provided. Figure 2 shows how the symbol “ordered-pair” can be introduced and given a common name in two different languages. It is to be noted that a symbol declaration is not a definition. One or more definitions can be given for a symbol.

The **OMDoc** DTD provides structural items: mathematical *concepts* such as **definitions** or **theorems**; and further items such as **examples**, **exercises**, and elaborative texts.

These items may contain formal elements (**FMP**), textual elements (**CMP**), metadata (**metadata**), and references (**ref**). A **CMP** is mainly a textual element that may contain objects (**OMOBJ** elements) and references. It is possible to refer to any URI, among others to concept identifiers and URLs. An **FMP** consists of one and only one **OMOBJ**. Figures 1 and 9 show an example of how a complex **OMDoc** item is built from basic elements.

The **OMDoc** in Figure 1 can be presented as follows:

If G is a group with unit e and $g \in G$, then the order of g is the smallest positive integer m with $g^m = e$. If no positive integer m with $g^m = e$ exists, we say that the order of g is infinite.

As in this example, every mathematical expression is built of symbols (the **OMS** elements) such as “in” and “eq”, of variables such as g or G (the **OMV** elements) and of applications (the **OMA** elements) of symbols and variables to arguments.

```

<definition id="def_order" for="order" type="simple">
  <metadata>
    <title xml:lang="en">
      Definition of the order of a group element
    </title>
    <extradata>
      <field use="mathematics"/>
      <abstractness level="neutral"/>
      <difficulty level="easy"/>
      <learning-context use="university_first_cycle"/>
      <depends-on>
        <ref theory="Th1" name="group"/>
        <ref theory="elementary" name="positive_integer"/>
      </depends-on>
    </extradata>
  </metadata>
  <CMP xml:lang="en" verbosity="3"> If
    <OMOBJ><OMV name="G"/> </OMOBJ> is a
      <ref xref="Th1_def_group">group</ref> and
    <OMOBJ><OMA><OMS cd="set1" name="in"/>
      <OMV name="g"/> <OMV name="G"/>
      </OMA></OMOBJ>,
    then the order of <OMOBJ><OMV name="g"/></OMOBJ>
    is the smallest positive integer
    <OMOBJ><OMV name="m"/></OMOBJ> with
  <OMOBJ id="OMOBJ_o1">
    <OMA><OMS cd="relation1" name="eq"/>
    <OMA><OMS cd="Th1" name="power"/>
      <OMV name="g"/>
      <OMV name="m"/>
    </OMA>
    <OMS cd="Th1" name="unit"/>
  </OMA></OMOBJ>.
  If no positive integer
  <OMOBJ><OMV name="m"/></OMOBJ> with
  <OMOBJ xref="OMOBJ_o1"/> exists, we say that the
  order of <OMOBJ><OMV name="g"/></OMOBJ> is infinite.
  </CMP>
</definition>

```

Figure 1. OMDoc representation of a definition of *order of a group element*

```

<omdoc>
  [...]
  <theory id="ida_elementary">
    [...]
    <symbol id="ordered-pair">
      <commonname xml:lang="en">
        ordered pair
      </commonname>
      <commonname xml:lang="de">
        geordnetes Paar
      </commonname>
    </symbol>
    [...]
  </theory>
  [...]
</omdoc>

```

Figure 2. A symbol definition in OMDoc

2.2.2. Metadata in OMDoc

For mathematical documents (including learning documents) it is not sufficient to use a representation that encodes the structure of mathematical documents and semantics of symbols. Additional information *about* the knowledge is needed in order to provide item types, bibliographical, and legal information, as well as relations between the items.

Therefore, the OMDoc DTD contains a `metadata` element compliant with the Dublin Core Metadata Element Set, Version 1.1. It contains the elements `contributor`, `creator`, `translator`, `subject`, `title`, `description`, `publisher`, `date`, `type`, `format`, `source`, `rights`, `language`, `relation`, `coverage`, and `identifier`.

The rest of metadata has been developed over time and it has historic reasons that currently, these metadata are contained in `extradata` as shown in Figure 1. Essentially, `extradata` is, currently, the slot that can contain elements of extended DTDs.

Also for historic reasons meta-information is also encoded in the `for` and `type` attributes of items currently. For example, the `for` in a definition contains a reference to the object that it defines and an assertion can have a `type="corollary"`, etc. This information is in attributes of the item as they have a logical meaning (for example the fact that a proof is *for* an assertion). In the near future, this information will be part of `metadata`.

2.2.3. *Additional Metadata in ACTIVEMATH*

For an adaptive learning environment, additional – in particular pedagogical – metadata are necessary. Therefore, we have extended the OMDoc DTD by metadata that express pedagogical properties and relations. Some of them are domain (mathematics) dependent, others are not.

The OMDoc in Figure 1 contains the pedagogical information **field**, **abstractness**, **difficulty**, **learning-context**, and **relation**.

field describes the field to which the content of the item belongs. It enables the system to provide items from particular fields (such as psychology, physics, or economy) if this is required by a pedagogical rule.

The meaning of **abstractness** and **difficulty** is clear, they serve to adapt the document to the skills of the learner. Currently, their ranges have three different values.

learning-context specifies for which learning context the material was intended originally. This information is important, when material from different sources is used within one course. The possible values of **learning-context** are the ones defined in the metadata standards from IMS and IEEE (see, for example [9]).

relation points to an item to which the given item is related to. The type of this relation is further specified in the **type**-attribute which can have the following meanings:

- the required previous knowledge to understand the item
- the similarity between an item and another one
- complementary for-relationships. (for instance (with **proof-for** and **example-for**, an item that is a proof for a theorem could as well be an example for a method application or an example could be a counterexample for a concept.)

Furthermore, the information about the verbosity of the textual parts allows the generation of different document types such as slides, summary, or a more detailed book. The CMP in Figure 1 has a **verbosity**-attribute.

For exercises, additional metadata serve to describe

- the technical type of the exercise such as name of the external system, multiple choice, etc.
- the pedagogical goal level such as knowledge, comprehension, application, or transfer. Such mastery-levels are distinguished in instructional pedagogy.

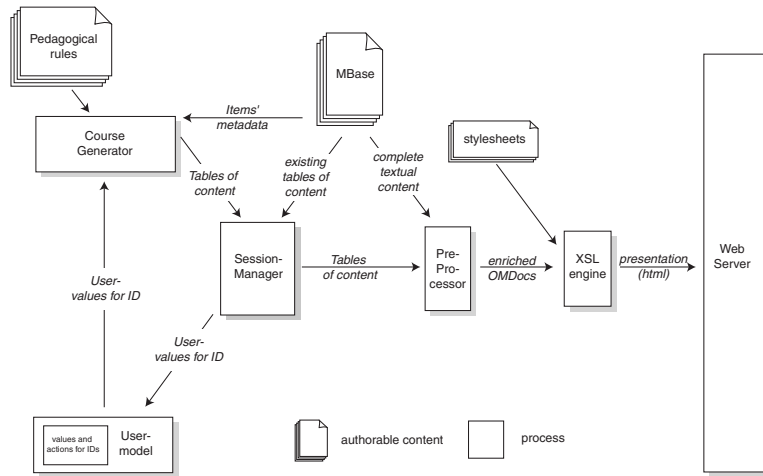


Figure 3. The architecture of the ACTIVEMATH server

- the task of the learner such as calculation, explore, prove, or model.

3. Knowledge Handling in ACTIVEMATH

The ACTIVEMATH server software has been designed to adaptively serve OMDocs using a Web-browser. This section describes how ACTIVEMATH manages extended OMDocs to present content and interactive exercises.

ACTIVEMATH has several components connected via the XML-RPC protocol [7]. The distributed architecture makes ACTIVEMATH easily extensible and, because of clearly defined interfaces, software components can be replaced or extended.

Figure 3 depicts the overall architecture of ACTIVEMATH. For the sake of simplicity, this architecture does not depict the exercise components, the updating of the user model, evaluators, and suggestion mechanisms. The figure contains the type of data handled by each component. It also shows which part of the OMDocs is manipulated where. Only MBASE, the pre-processor, and XSL-stylesheets handle OMDoc's complete textual content (the children of the CMP elements which include formulae). The FMP children are only used by proxies that connect to mathematical systems. Whereas most of the other compo-

nents handle references to `OMDoc` items by means of unique identifiers and in some cases additionally handle the metadata.

The session manager is the central state-storage mechanism that allows to keep the different 'books' for a user, pre-made books as well as dynamically generated books. (see Section 3.4). Each stored table of content contains a hierarchy and the references to the items on a page.

The course generator is the component that chooses the content to be presented to the learner. It is described in Section 3.4.

Although this architecture appears to be relatively general for a server-architecture, its underlying programs are specialized for `OMDoc` and mathematical content. The objects manipulated that represent XML elements are all based on a library specially designed to access `OMDoc` information (the `OmdocJdom` library which is developed independently of the server). Moreover, the course-generator is currently designed for mathematical courses, that are structured to proofs, assertions, definitions and their relations to other items.

To prepare the delivery of a content page, the pre-processor is invoked to produce a page. First, it replaces identifiers by the complete textual content of each item from `MBase` as well as its metadata. Then it processes the assembled `OMDoc` to prepare it for the next step and make interactive exercises accessible. Finally, an XSL-style sheet is applied to produce a format that the browser can read (currently HTML and PDF produced by \LaTeX).

3.1. MATHEMATICAL CONTENT PRESENTATION

This section explains in more detail how the rendering is performed, how it can be extended, and how interactivity and adaptivity can be packed with it.

Converting a content "on the fly" for the delivery is at the heart of ACTIVE MATH content presentation. It allows to keep the content in `MBase` and to flexibly adapt the presentation to the user's requirements and preferences.

Before the content is delivered to the browser, the `OMDoc` is converted to an appropriate presentation-media language by the pre-processor and XSL-style sheets.

3.1.1. *Viewing Mathematical Content on the Web*

As mentioned in the Status Report of Design Science Inc. [12], the majority of higher level mathematical content is encoded using one of the flavors of \TeX whereas lower level mathematics tends to be written with classical word-processors. The readable documents are is

generally distributed as Adobe Inc.'s Portable Document Format (PDF) or PostScript (PS).

ACTIVEMATH is a learning environment and therefore envisions content presentation as an interactive experience rather than these print-formats. For this reason, at the possible expense of sacrificing a part of mathematical formulae display, ACTIVEMATH presents its content in HTML with Unicode characters to display mathematical symbols currently.

The quality of the HTML presentation is recognized to be still imperfect because

- HTML is less than appropriate for fine-grained layout of text containing mathematical formulae
- the support for Unicode characters display is extremely dependent on the browser version, operating system, etc. Consequently Unicode support can only be tested visually by the user herself.

The display of mathematical content on the Web has always been a poor child within Web-browsers. This is especially true compared the quality of the layout of T_EX engines, a quality that is still unsurpassed. To remedy this problem, the MATHML-presentation language [2], describing the layout of mathematical formulae for dedicated viewers, has been developed. Now some browsers start to support it in the form of embedded MATHML-presentation elements within XHTML content. An XHTML+MATHML of ACTIVEMATH is being worked on.

The further development of presentation processes in ACTIVEMATH follows a delivery based on the T_EX layout. A PDF-presentation used as a print-version has been recently added. Other target-formats are also being prepared, among others, a L^AT_EX output converted to FLASH or SVG will be able to combine high-quality layout with the rich interactivity of these formats.

3.1.2. *Converting Content to Presentation*

Before one of the output formats can be produced, the OMDoc-encoded content collected from MBASE has to be converted. The XML stylesheet language, XSL transformation [4], is well-suited for this task. It processes XML-documents via templates to return any form of textual or non-textual data.

Although the conversion process is rather resource-intensive, it realizes a great amount of flexibility and this is an important step towards simple re-usability. It is also a basis for future alternative media presentation for which no presentation markup exists yet.

The separate handling of the presentation eases the constantly ongoing adaptation to changes in browsers. We experienced this fact during

```
<presentation for='boundary' parent='OMA' fixity='prefix'>
  <use format='TeX'>\partial</use>
  <use format='HTML'>&#x2202;</use>
</presentation>
```

Figure 4. Presentation tag for the mathematical symbol with identifier “boundary” for the output targets HTML and TeX.

the development of ACTIVE MATH which can now successfully display on Mozilla ($\geq 0.9.8$) and Internet Explorer (≥ 5.1). Several times faulty behaviors were observed in these browsers. The corrections affected only the presentation engine and not the content.

Since new OMDoc symbols can be declared at any place in the content, the stylesheets have to be extensible and to allow the author to enrich them with informations on how to present this symbol. This extension capability is very rarely seen in other presentation engines. It is not available in engines such as the WebEQ and MathType families of products.⁴

Aside of the purely semantic encoding of content, OMDoc allows for some presentational information in the `presentation` element that declares how a symbol should be presented in each target output. Figure 4 illustrates this. This declaration is the way to specify a particular presentation. A simple process produces an XSL-template from the `presentation` elements of OMDocs (as in Figure 4). This template is then inserted into the stylesheet. An example of template that renders a symbol is in Figure 5. This template is a separate part of the stylesheet and can be added and modified at any time to extend the set of viewable symbols.

```
<xsl:template
  match="OMA[OMS[position()=1
  and @name='boundary' and @cd='topDiff_intro']] ">
  <xsl:text disable-output-escaping="yes">&#x2202;</xsl:text>
  <xsl:apply-templates select="*[2]" />
</xsl:template>
```

Figure 5. The XSL-template for the presentation for the symbol “boundary” generated in HTML is generated from the `presentation` tag of Figure 4.

3.1.3. *User-Adaptivity in Presented Documents*

The user-adaptivity of a document presentation is a feature for which the Web has a lot to offer and where the technology can become quite elaborate. By adaptivity, we mean the entire set of features which adapt the document presented in order to follow the user's profile and to react to actions of the user.

What has a structured semantic encoding to offer for user-adaptivity?

The representation of the content consists of units that contain text and abstract elements. In this setting, the introduction of a new adaptivity feature is not as simple as writing a handler for an element (or using buttons and menus) as is done for HTML. Instead, the design of a feature involves modifying XSL-templates and/or pre-processor tasks that will generate the presentation code. Therefore, the representation seems pretty restrictive at the beginning because no explicit presentation information is given. The advantage comes on the long run, when previously encoded content is used in an updated environment. A new adaptivity feature is introduced by modifications of the presentation engine or its data whereas, for HTML, the same task requires an extensive manual work as every single page would have to be edited manually.

In ACTIVEMATH the adaptivity is realized by the course-generator, the pre-processor and the stylesheet-transformation. These software components are cleanly isolated and can be enriched easily by a developer. Currently, ACTIVEMATH provides the following adaptivity:

- The language is picked according to the user's choice.
- The user's style preferences determine a personalized presentation.

The personalization includes the choice of appearance and language. Another facet of personalization is the adaptation to different mathematical notations for the same notion. It is known that mathematical notations around the world vary. For example, the way the English mathematical culture writes the compositions and applications of maps differs from the common usage in other countries. Another example is the Polish notation for quantifiers. An adaptation of the presentation of mathematical formulae to different cultures is possible thanks to the "universal language" that the `OpenMath` encoding provides which is translated to an appropriate graphical notation.⁵

Other features enhance the interactive flavor of ACTIVEMATH's documents, namely:

- URLs are encoded to respond to user's clicks

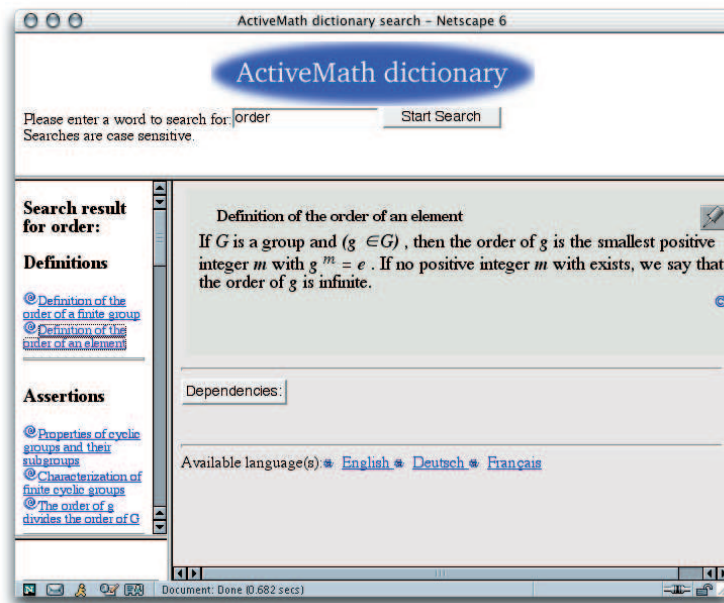


Figure 6. Presentation of the item “Definition of an Euler indicator” in the dictionary

- When the mouse flies over the presentation of a mathematical symbol, its name (in the appropriate language) is displayed in the *status bar* of the browser.
- A click on the symbol’s presentation opens the dictionary, which then shows the description of the symbol and provides links to its definitions and occurrences.

The last two features, are consequences of the presentation processes in ACTIVE MATH. They offer a quick access to the appropriate information to explain the meaning of symbols. They are important for the publication of mathematics on the Web because it may help the user to grasp the notational framework. For instance a user browses a new mathematical knowledge repository which uses notations that are foreign to her. To be able to grasp the meaning of a new symbol or simply remember it, she may have to browse through several pages of this repository. In a traditional book, a glossary may help. In a hypertext, jumping to the glossary pages (if available at all) may confuse the user and end-up in navigation problems.

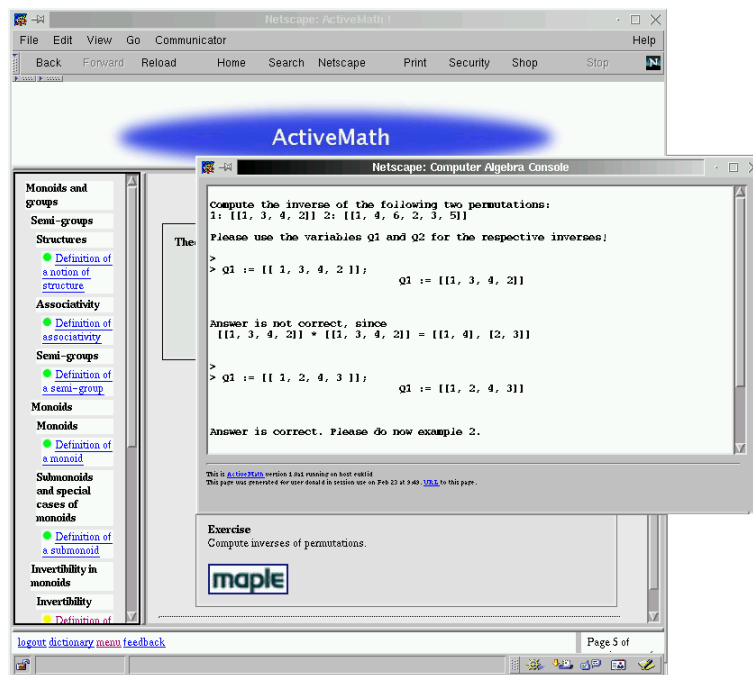


Figure 7. A Computer-Algebra console interacting with a MAPLE engine on the server.

3.2. DICTIONARY

The ACTIVEMATH dictionary presents content in the simplest form. An example is shown in Figure 6. Units are presented one at a time together with their relations to other items. These relations include their occurrences in other items, the items they depend on, and vice versa. This way of presenting content supports mathematicians which are looking for a simple way to organize their knowledge and the relationships. Currently, the dictionary can be browsed by a full-text search. In the future, it will also be able to search through mathematical formulae and restrict the type of items it will display.

3.3. INTERACTIVITY: EXERCISES WITH EXTERNAL MATHEMATICAL SYSTEMS

As a way to promote active learning, ACTIVEMATH goes further than presenting content. Like most learning environments ACTIVEMATH has a multiple-choice-question exercise type but in addition, it offers interactive exercises with external systems. Currently, three types

```

<exercise id="examplesandexercises_Exc1" type="for"
  for="c6s4p1_Th1_def_group" assertion="INVERSE-Z3-PLUS">
  <metadata>
    <extradata>
      <difficulty level="easy"/>
      <abstractness level="neutral"/>
      <depends-on>
        <ref theory="c6s1p4_Th2" name="monoid"/>
        <ref theory="c6s4p1_Th1" name="inverse"/>
        <ref theory="c6s4p1_Th1" name="group"/>
      </depends-on>
    </extradata>
  </metadata>
  <CMP format="omtext" xml:lang="en">Each element of the monoid
    [...] therefore it is a group.
  <omlet argstr="examplesandexercises_Exc1_code1" function="loui"/>
  </CMP>
</exercise>

<code id="examplesandexercises_Exc1_code1">
  <data>
    <startup>
      <command>DECLARE-ASSERTION</command>
      <param><replace xref="INVERSE-Z3-PLUS"/></param>
    </startup>
    <startup>
      <command>MULTI-PLANNER-INTERACTIVE</command>
      <param>(REDUCETOSPECIAL EQU SOLVE TRYANDERROR)</param>
    </startup>
  </data>
</code>

```

Figure 8. An Ω MEGA exercise that contains the invitation to perform the exercise. This invitation, when clicked by the user, invokes the related code element which instructs Ω MEGA to start an interactive proof of the assertion encoded in Figure 9.

of interactive exercises are supported: MAPLE and MUPAD exercises in which the CAS is used from a console as depicted in Figure 7, and exercises with the proof planner Ω MEGA that use the L Ω UI user interface.

The representation of an exercise generally includes two parts: an initialization code defining variables, help functions, etc. and a code that checks the user's solution. For example, the initialization loads libraries and the solution-checking code must perform conditional actions. Hence, they cannot simply be encoded using `OpenMath` or `OMDoc`. However, these codes can include `OMDocs` for mathematical formulae and references to formal theories. These statements are converted to the

```

<assertion id="INVERSE-Z3-PLUS" type="conjecture" theory="ZMZ">
  <CMP xml:lang="en">
    The structure
      <OMOBJ><OMA><OMS cd="ZMZ" name="RESCLASS-SET"/>
        <OMI>3</OMI></OMA>
        <OMS cd="ZMZ" name="PLUS-RESCLASS"/>
        <OMA><OMS cd="ZMZ" name="RESCLASS"/>
        <OMI>3</OMI><OMI>0</OMI></OMA>
      </OMOBJ>
    is a group.
  </CMP>
</FMP>
<conclusion id="INVERSE-Z3-PLUS-CONC">
  <OMOBJ>
    <OMA>
      <OMS cd="STRUCT" name="INVERSE-EXIST"/>
      <OMA><OMS cd="ZMZ" name="RESCLASS-SET"/>
      <OMI>3</OMI></OMA>
      <OMS cd="ZMZ" name="PLUS-RESCLASS"/>
      <OMA><OMS cd="ZMZ" name="RESCLASS"/>
      <OMI>3</OMI><OMI>0</OMI></OMA>
    </OMA>
  </OMOBJ>
</conclusion>
</FMP>
</assertion>

```

Figure 9. An OMDoc assertion stating that the structure $(Z_3, +, 0)$ is a group.

system's native input representation. Currently, this only works with the Ω MEGA proof planner (an XSL style-sheet translates the OMDoc theories to the Ω MEGA language). For the CASs the freely available *phrase-books* which exist for several systems could not yet be used because of their low quality.

The initialization code for an exercise in the Ω MEGA proof planner shown in Figure 8. It contains the LISP instructions used to load the necessary resources. The element referred to by `replace` will, on serving time, be converted to the Ω MEGA-specific input language. One advantage of this parametrization is that only the part specific to a system needs to be rewritten when several mathematical systems are used.

3.4. COURSE GENERATION

The course generator is responsible for choosing and arranging the content to be learned. It contacts the mathematical knowledge base, MBASE, in order to calculate which mathematical concepts are re-

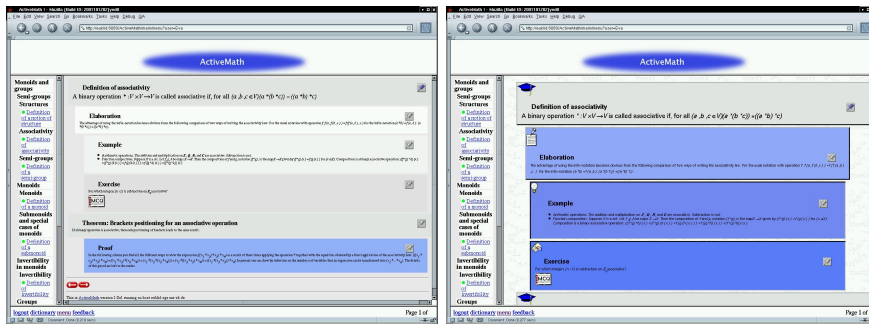


Figure 10. Two generated courses with the same goals generated for two users with different estimated skills and different preferences

quired for the learner to understand the goal concepts that the user has chosen, checks the user model in order to find out about the user’s prior knowledge and preferences, and uses *pedagogical rules* to select and arrange the content in a way that is suitable for the user.

The course generator makes extensive use of the metadata information:

- Retrieval: Currently, the retrieval of the mathematical content uses the *depends-on* and *for* dependencies. *depends-on* is needed to retrieve all concepts necessary to understand the goal concepts; *for* is used to retrieve additional items related to them. For example, exercises, examples, or elaborating texts for a concept are fetched.
- Selection and order: Not all of the retrieved content is useful for every learner, e.g., exercises that require the use of Ω MEGA will more likely confuse than help learners that have never seen Ω MEGA before. Therefore, pedagogical rules decide which content to choose for which learner. These rules use the ACTIVEMATH metadata *difficulty*, *abstractness*, and *field*. For example, one rule determines that if the user has chosen a detailed guided tour and has a high knowledge of a definition, then an introduction and the definition itself is presented, and in addition a difficult example and a difficult exercise. Another rule states that if the user’s knowledge of the definition is low, then a motivation, three exercises and three examples with increasing difficulty should be presented.

Figure 10 shows what a generated course looks like for two users with different knowledge and preferences.

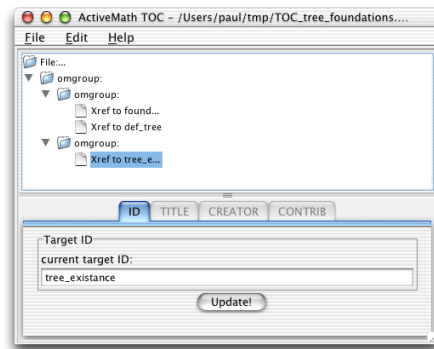


Figure 11. The table of content editor used to assemble a book from existing content.

3.5. RE-ARRANGING CONTENT ITEMS

The fact that items can be re-arranged and combined in different orders is not familiar for most authors. Chapter numbering, linking sentences, and word references like “*as we have seen in the previous section*” do not work anymore. Instead, references by name and meaning have to be used. This is, however, not new for hypertext documents that can be browsed in any order.

In ACTIVEMATH, the selection of content, the *editor* task, can be performed with a visual tool for editing tables of content. A screen-shot is depicted in Figure 11. Items can be introduced into a book by drag-and-drop either from a course page, from another table of contents, or from the dictionary. The tool can also import already existing tables of contents from previously generated courses.

The work of re-arranging the items’ order and packaging them in a book is a task that differs from the actual authoring task. We expect teachers to perform it, when they are preparing a course from existing content collections.

The table-of-contents editing is possible because of the ‘context-free’ character of the knowledge representation: OMDoc documents are not bound to external ‘library files’ like L^AT_EX is. The only connections an OMDoc document has to other documents are the references to items in other documents or collections. As a result, OMDocs are easily exchanged and shared.

4. Authoring

Authoring includes writing the textual content, declaring new symbols, writing mathematical formulae, and writing exercises. The expected result is a valid collection of OMDocs.

```

Definition:[<-def_demi_espace]
:"demi-espace"
:for:[demi_espace]
On appelle demi-espace et l'on note  $H_n$  l'ensemble des points de  $R^n$ 
dont la dernière composante est positive:
 $H_n = \text{set}(x \mid x \in R^n \wedge \pi(n,x) = 0)$ 

```

Figure 12. An extract of a QMATH source text.

Currently, an elementary validation is performed using the XML validation against the OMDoc and ACTIVEMATH document type definitions. This validation checks the appropriate nesting of elements and the missing attributes, which is enough for the most frequent errors. Another validation tool the OmdocJdomMBase, has been developed to manipulate a collection of OMDocs, it reports errors for wrong references. These validations do not, however, support types of mathematical objects and their possible application conditions; such an elaborate validation that is possible because of the semantic encoding is still future research.

To find symbols with the appropriate semantic, authors may have to browse through symbol lists. The OpenMath content dictionaries provide such a list, in which all the symbols are grouped by content dictionary and described by examples. These *core* content dictionaries of OpenMath form a well defined foundation. As we mentioned in paragraph 2.2.1, additional content dictionaries can be defined anywhere in the content. Browsing the ACTIVEMATH's dictionary can support authors to find the right symbols.

Currently, the tool for authoring OMDocs, in particular formulae, is QMATH [8]. QMATH has an line-based input syntax. For mathematical formulae, the QMATH syntax is close to the syntax of computer algebra systems. An example of QMATH source is depicted in Figure 12. Note the use of mathematical symbols such as the greek letters and the conjunction symbol \wedge which is made possible by the Unicode support of QMATH.

5. Roles for Developing ACTIVEMATH Content

Let us summarize the roles involved in the content creation processes.

- the author is responsible to write valid OMDoc collections
- the editor is responsible for preparing structured courses with existing content
- the presentation developer is responsible to maintain the presentation process and its associated data

- the author of pedagogical rules modifies and writes generic pedagogical patterns to be used in the course generation
- the exercise developer is responsible for implementing the user-interface, its behaviors, and, possibly, the connection mechanism to the mathematical system, following the author’s instructions.

These roles can be separated. As an example of good separation of the roles, the presentation developer interacts with the author only to obtain the presentation instructions for the symbols. No further interaction is needed because the validators are rich enough to check the completeness and correctness of the authored content. The separation is less clear between the author and exercise developer as the description of user interfaces and their behavior cannot be as clearly defined as semantically encoded.

With the current status of the available tools –although still rudimentary– we managed this task separation successfully. A developer and a mathematics professor were brought together to start the creation of a content in statistics. After training and tool debugging, the professor can now write her content independently of the developer who is now only responsible for the presentation processes and the development of exercise applets.

6. Comparable Solutions

The semantic encoding, together with the interactivity and adaptivity features make of ACTIVEMATH a solution incomparable to most existing software publishing mathematical content.

There are major differences between publishing mathematical content with the ACTIVEMATH server and publishing presentation-oriented mathematical content like the emerging XHTML+MATHML, or the classical L^AT_EX generating DVI or PDF formats. The following features cannot be offered by XHTML+MATHML and L^AT_EX-based languages: the ability to present in several viewer-formats, the ability to re-arrange content-items without changing the source documents, the ability to personalize the presentation of mathematical formulae, the ability to enrich the presentation with interactive features that support the user, the complete searchability, and the ability to share content between presentable and mathematical systems’ content. classical Web design, a modularization of the content and the setup of abstract features can realize the mentioned features. Macro-based languages, such as L^AT_EX, the numerous script-in-page languages for HTML (including ASP, JSP, PHP) offer ways to modularization. Template-languages, such as the

WebMacro or JSP tag-libraries offer similar facilities. In addition, they separate the content from programs. With them, some of the features listed may be reached.

ACTIVE MATH knowledge representation goes further than template languages in that it only allows a limited encoding. This limitation guarantees many re-usability features. Moreover, the mathematics-specific character of the representation guarantees machine-readability.

Three projects are worth mentioning as they somewhat resemble the ACTIVE MATH approach:

Slicing Information Technologies [6] products, developed by the University of Koblenz-Landau and now sold commercially use the \LaTeX encoding, cut into slices which are annotated with metadata. The content is presented in the PDF format with the help of the `pdflatex` program. A choice of content according to the user's wishes and simple maybe incoherent combination of presentations are features that can be performed. Interactive exercises and an updateable user model are missing. The choice of the \LaTeX language for the encoding of content presents severe limitations which ACTIVE MATH has avoided: no adaptive presentation can be achieved and sharing between exercises and content is impossible.

The emerging MATHBOOK encoding [5] developed in the Technical University of Eindhoven appears to have promising objectives which are comparable to the ones of ACTIVE MATH. MATHBOOK may offer more flexibility than OMDoc as it is an extension of DOCBOOK, a rather rich book-publishing encoding. MATHBOOK is enriched with `OpenMath` objects and JSP-tags developed by the RIACA group in Eindhoven. We believe that OMDoc is simpler and stricter, and more appropriately defines the essential building blocks of a mathematical knowledge.

The Hypertext Electronic Library of Mathematics, HEAM [1], is a project for the Web publishing of formal mathematical content. The project proposes to spread mathematical content encoded in a MATHML-content extension through simple Web-servers. It also offers XSL-style-sheets for presentation and generic interfaces to mathematical software that could process the formal language. The general objective of content exchange that would be suitable for mathematical programs meets the goals of OMDoc. However, currently, the HEAM library requires the use of a verbalizer for presentation, a tool that is still at the research level. OMDoc and ACTIVE MATH avoid this problem because they include both, verbal and formal content.

7. Future Work

Currently, a suggestion mechanism, an improved MBASE, and a visual authoring tool are being developed.

The MBASE queries currently used by the ACTIVEMATH server are almost only elementary extraction queries. However, this database engine already processes queries such as “find all the statements of commutativity”. Such queries can be put to good use in the authoring process and will support more elaborate management actions such as the detection of duplicates, the comparison between content collections, and the search for appropriate items by an editor.

On the server-side, one of the new features is the enrichment of the presentation with extra information allowing the user to select sub-terms of mathematical formulas. This is a basis for copy-and-paste between content presentation and exercise user interfaces which will involve the use of the phrasebooks converting *OpenMath* content to mathematical systems’ specific languages. The phrasebooks are developed by other universities and we expect to be able to use them soon.

A visual authoring tool will support the authoring of metadata through intelligent analyzers that suggest to the author values that seem appropriate. This visual tool will provide an editable view of the current content, including a formula editor and metadata annotations. The tool will gradually integrate all facets of the authoring process such as the edition and debugging of exercise content, of pedagogical rules, and book’s tables-of-contents. It will also be the basis for related content manipulation tools such as the support for merging content collections that have intersections, or for the implementation of a center-point for simultaneous connections to mathematical systems.

In the near future *OMDoc* and *ACTIVEMATH* will become compliant to emerging standards such as, for educational purposes, the IMS and related IEEE standards.⁶

8. Conclusion

As a real mathematics education system, *ACTIVEMATH* brings together important approaches from different communities: Web-publishing and -authoring, representation of mathematical knowledge, and intelligent tutor systems.

ACTIVEMATH differs from other intelligent learning environments in many aspects. In particular, it represents its content in a semantic XML-language for mathematics, annotated with pedagogical metadata.

Moreover, the separation of the content from its presentation is strictly realized in ACTIVEMATH. Some of the resulting advantages are:

- The reuse of content for different purposes and different contexts is possible.
- The content can be searched in a more precise fashion than any presentation-oriented encoding. In particular, formulae can be searched.
- One may flexibly choose between different ways to present consistently a mathematical symbol or element.
- Different kinds of presentation such as L^AT_EX output, HTML output or simple text in a console applet can easily be generated.

From the simple application of an XSL-style sheet to a truly dynamic server that adapts the presentation and offers interactivity, the ACTIVEMATH development has taken more than two years. It will, eventually, be distributed in an Open-Source fashion. OMDoc had to be extended for several purposes, more often than not initiated by author demands. For example, the multiple-choice-questions elements have been too inflexible.

The usability of ACTIVEMATH from the point of view of a student user has been an important focus. Usability studies have been made and classroom exercise sessions have taken place. Although these experiments have been preliminary, they have raised quite a number of issues to allow the best learning experience. Adjustments to the system and presentation process could be performed with almost no modifications to the existing content.

Acknowledgments

Our students have contributed to several functionalities of ACTIVEMATH described in this paper, in particular, Adrian Frischauf worked on the integration of external systems, Eric Andres on the suggestion mechanism, and Thomas Sewenig implemented a table-of-contents editor, Martin Sunkel worked on the presentation process, and Martin Fuchs worked on the database connection.

We would also like to thank Martin Pollet, Vladimir Brezhnev, and Immanuel Normann for fruitful exchanges.

Notes

- ¹ An ACTIVEMATH demo is available at <http://www.activemath.org/demo>
- ² <http://www.w3.org/2001/sw/>
- ³ About OpenMath, the content dictionaries and the phrasebooks, please see their web-site <http://www.openmath.org>.
- ⁴ <http://www.dessci.com/webmath/>
- ⁵ As ACTIVEMATH presents content on the Web, internationalization is a natural step which was performed. OMDoc representation allows the CMP elements to have a language attribute, so that items can be written in several languages. Moreover, the messages provided by the ACTIVEMATH server also been internationalized. ACTIVEMATH currently speaks English, German, French, and Russian.
- ⁶ Most of the e-learning related standards are discussed in IEEE forums and developed by the IMS consortium. A good starting point is the IMS home page: <http://www.imsglobal.org>.

References

1. Asperti, A., L. Padovani, C. Coen, and I. Schena: 2001, 'HELM and the Semantic Web'. In: R. J. Boulton and P. B. Jackson (eds.): *Proceedings of the 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2001)*, Vol. 2152 of LNCS. pp. 59–74. See also <http://www.cs.UniBo.it/helm/>.
2. Carlisle, D., P. Ion, R. Miner, and N. Poppelier: 2001, 'Mathematical Markup Language, version 2.0'. <http://www.w3.org/TR/MathML2/>.
3. Char, B., G. Fee, K. Geddes, G. Gonnet, and M. Monagan: 1986, 'A Tutorial Introduction to MAPLE'. *Journal of Symbolic Computation* **2**(2), 179–200.
4. Clark, J.: 1999, 'XSL Transformation'. World Wide Web Consortium, see <http://www.w3.org/Style/XSL/>.
5. Cuyppers, H. and H. Sterk: 2001, 'MathBook, Web-technology for Mathematical Documents'. In: *Proceedings of the BITE 2001 conference*. See also from <http://www.riaca.win.tue.nl/>.
6. Dahn, I.: 2001, 'Slicing Book Technology - Providing Online Support for Textbooks'. In: *Proceedings of the ICDE 2001, International Conference on Distant Education*.
7. Dave Winer: 1999, 'XML-RPC Specification'. <http://www.xmlrpc.org/spec>.
8. González Palomo, A.: 2002, 'QMath: An Authoring Tool for OMDoc'. See also <http://www.matracas.org/> and section 4.2 of OMDoc 1.1 specification (from <http://www.mathweb.org/omdoc/>).
9. IEEE Learning Technology Standards Committee: 1999, 'Learning objects metadata'. Available from <http://www.edna.edu.au/edna/aboutedna/metadata/analysis/LOM.htm>, see also <http://ltsc.ieee.org/>.
10. Kohlhase, M.: 2001, 'OMDoc: Towards an Internet Standard for Mathematical Knowledge'. In: E. R. Lozano (ed.): *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2000*. See also <http://www.mathweb.org/omdoc>.
11. Melis, E. and J. Siekmann: 1999, 'Knowledge-Based Proof Planning'. *Artificial Intelligence* **115**(1), 65–105.

12. Miner, R. and P. Topping: 2002, 'Math on the Web, a Status Report: Focus Authoring'. Technical report, Design Science Inc. Available at http://www.dessci.com/webmath/status/status_Jan_02.stm.
13. Schönert, M.: 1995, 'GAP – Groups, Algorithms, and Programming'. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany.
14. Sorgatz, A. and R. Hillebrand: 1995, 'MuPAD - Ein Computeralgebra System I'. *Linux Magazin* (12/95).
15. Wolfram, S.: 1999, *The Mathematica Book*. Cambridge University Press, fourth edition.

