OpenMath Technology for Interactive Mathematical Documents

Olga Caprotti, Arjeh M. Cohen, Hans Cuypers, and Hans Sterk

Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

1 Introduction

Emerging new technologies as XML, XSL and both MathML and OpenMath make it possible to bring mathematics to the Internet. Indeed, OpenMath, a markup language for mathematical content, and OmDoc, its extension to mathematical documents, open a way of commuting mathematics between computers, between software applications and over the internet without losing information. In this paper we describe the latest applications of OpenMath related technologies for Interactive Mathematical Documents. As an example we describe the way we incorporate these new technologies in a new version *Algebra Interactive*, an interactive course on first and second year university algebra.

2 OpenMath technologies

MathML and OpenMath are the predominant markup languages for mathematics [**?**]. MathML2 [**?**] cleanly defines the interrelations between MathML and OpenMath. Ideally these two languages complement each other: MathML presentation can be used for presenting mathematical content written in OpenMath. In fact, OpenMath is a language for the representation of mathematical content only. The document that describes the OpenMath standard is [**?**].

Yet, OpenMath alone is not a natural choice for expressing a corpus of mathematical knowledge because it lacks the mechanisms to relate concepts, for instance in hierarchies made of definitions, theories and theorems. The OpenMath Document Specification (OmDoc) [**?**], currently under development by Kohlhase, is an XML document type definition that can be used to represent general mathematical knowledge the way is written in lecture notes and in scientific articles, but also in mathematical software like algebraic specification modules or library files of a proof checker. It is being used as source format for the next release of *Algebra Interactive!* [**?**], an interactive textbook used in teaching first year university algebra. OpenMath Documents are moreover intended as the input format for Mbase [**?**], a knowledge base of mathematics.

In this section we briefly discuss the OpenMath language, its extension to OmDoc and some OpenMath techniques relying on OpenMath and OmDoc.

2.1 OpenMath and OpenMath Documents

The OpenMath standard language [**?**] provides a semantically rich representation of mathematical information for electronic access and usage. Here we limit the exposition on OpenMath to some examples of mathematical objects that occur later in the paper. The reader is referred to the standard documents available from [**?**] for the details.

Algebraic structures, like the polynomial ring $\mathbb{Z}_p[X]$, are representable as OpenMath abstract objects using **application** objects like:

$$\textbf{application}(\texttt{polyr:PolynomialRingR}, \textbf{application}(\texttt{setname2:Zm}, p), x) \quad (1)$$

OpenMath *Content Dictionaries* (CD) collect and provide definitions of mathematical notions for usage within OpenMath applications. The official repository for CDs is [**?**]. The OpenMath symbols in the object above that identify the polynomial ring structure obtained from the integers modulo p are `polyr:PolynomialRingR` and `setname2:Zm`. More precisely, they are the symbols called `PolynomialRingR` and `Zm` defined in the CDs `polyr` and `setname2`, respectively.
A polynomial in this ring, say $f = X^3 - X + 1$, can be represented in several ways as an abstract OpenMath object, for instance by using the `polyr:PolynomialR` constructor for recursive polynomials. As with all OpenMath objects, it can be encoded in a human-readable format using XML [**?**] and stored as:

```
<OMOBJ><OMA>
            <OMS cd="polyr" name="PolynomialR"/>
            <OMA>
                 <OMS cd="polyr" name="PolynomialRingR"/>
                 <OMA>
                      <OMS cd="setname2" name="Zm"/>
                      <OMV name="p">
                 </OMA>
                 <OMV name="x"/>
            </OMA>
            <OMA>
                 <OMS cd="polyr" name="PolyRrep"/>
                 <OMV name="x"/>
                 <OMA>
                      <OMS cd="polyr" name="monomial"/>
                      <OMI> 3 </OMI>
                      <OMI> 1 </OMI>
                 </OMA>
                 <OMA>
                      <OMS cd="polyr" name="monomial"/>
                      <OMI> 1 </OMI>
                      <OMI> -1 </OMI>
                 </OMA>
                 <OMA>
                      <OMS cd="polyr" name="monomial"/>
                      <OMI> 0 </OMI>
```

```
                    <OMI> 1  </OMI>
                </OMA>
            </OMA>
        </OMA>
</OMOBJ>
```

In this example, the outermost XML element `<OMOBJ>` encloses nested OpenMath application objects, appearing within the element `<OMA>`, which are built using OpenMath symbols (`<OMS>`), OpenMath variables (`<OMV>`), and integers (`<OMI>`). Notice that the application object highlighted by the box is essentially the XML encoding of the polynomial ring expressed abstractly in (1).

The OpenMath Document Specification (OmDoc) [**?**], is an extension of OpenMath for the communication of mathematical knowledge in a general sense including: papers, letters, interactive books, e-mails, talks, communication between mathematical/computational services and so on.

As a markup language, OmDoc supports elements for representing the accepted structure "definition, theorem, proof" used by many mathematicians in papers and books. These can be organized in "theories". Special attention has been devoted in particular to proofs and they can be markup up in a variety of ways to ensure no loss of semantical structure. For instance, OmDoc provides elements to identify steps in a proof, premises, conclusions, and methods. Every mathematical entity is glued by intermediate explanatory text and possibly auxiliary items like exercises, applets, and examples.

Here is a fragment of an OpenMath Document that defines in the natural way the binary predicate divides for natural numbers. Although the XML source is readable, it is not user-friendly at all.

```
<theory id="primes">
<definition type="inductive" id="div" item="divides">

<CMP format="omtext" xml:lang="en">A natural number <OMOBJ><OMV name="n"/></OMOBJ>
divides a natural number <OMOBJ><OMV name="m"/></OMOBJ>, denoted <OMOBJ><OMA>
<OMS name="divides" cd="ida"/><OMV name="n"/><OMV name="m"/></OMA></OMOBJ>,
if there exists a natural number <OMOBJ><OMV name="q"/></OMOBJ> such that
<OMOBJ><OMA><OMS cd="relation1" name="eq"/><OMV name="m"/><OMA>
<OMS cd="arith1" name="times"/><OMV name="n"/><OMV name="q"/></OMA></OMA></OMOBJ>.
</CMP>

 <FMP>
 <OMOBJ><OMBIND><OMS cd="lc" name="Lambda"/>
  <OMBVAR><OMATTR><OMATP><OMS cd="icc" name="type"/>
                  <OMS cd="setname" name="N"/>
  </OMATP>        <OMV name="n"/>    </OMATTR>
  <OMATTR><OMATP><OMS cd="icc" name="type"/>
                 <OMS cd="setname" name="N"/>
  </OMATP>       <OMV name="m"/>    </OMATTR>
  </OMBVAR>
  <OMBIND><OMS cd="quant1" name="exists"/>
  <OMBVAR><OMATTR><OMATP><OMS cd="icc" name="type"/>
                 <OMS cd="setname" name="N"/>
  </OMATP>       <OMV name="q"/>    </OMATTR>
  </OMBVAR><OMA><OMS cd="relation1" name="eq"/>
               <OMV name="m"/>
               <OMA><OMS cd="arith1" name="times"/>
                    <OMV name="n"/>
                    <OMV name="q"/>
               </OMA></OMA>
   </OMBIND></OMBIND>
   </OMOBJ>
 </FMP>
</definition>
...
```

The reader may skip to Figure **??** to see how it is presented when browsing with Netscape. The examples in the following sections will consider OpenMath objects appearing in this fragment.

For generality, the mathematics in OpenMath Documents is represented using OpenMath objects. Having the content in OmDoc format, we are left with the problem of how to produce high-quality presentation. Since the OpenMath and OmDoc objects do not implicitely carry any presentation directive, it is up to the user to decide the rendering mechanism.

****

Here XSL-style sheets to obtain presentation in HTML, LaTeX or whatever you want ....

2.2 OpenMath Servers

An OpenMath server is a server interfaced to a combination of mathematical back-engines. It handles requests invoking computational aspects of a mathematical nature, e.g. symbolic integration or proof verification via OpenMath Phrasebooks, programs interfacing the back-engines to the client via OpenMath.

A Phrasebook declares the list of OpenMath CDs it recognizes. Within this OpenMath setting it is able to perform several tasks, specified in the *control information*, like EVAL, SIMPLIFY, PROVE, SOLVE,

PRINT, and the *interpretation* of what is going to be done to the OpenMath objects using the back-engines. Interpretation is a function of the control information and the received OpenMath objects. Finally, the Phrasebook specifies how the actual *communication* between the software package and the OpenMath computer environment is achieved.

Although a Phrasebook can only function properly if it is specified right down to the nitty-gritty detail, there is as yet no guarantee that the actual implementation of the Phrasebook is conformant to its specification.

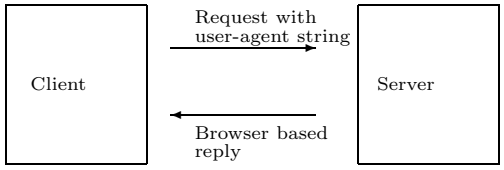
Upon input, which includes a choice of task to be performed, the Phrasebook invokes the module for encoding the expression as OpenMath object. This module also takes care of decoding an OpenMath object into a system-specific syntax. It is called `codec` and depends on the list of CDs for its translation (e.g. it might encounter a symbol not defined in its list of CDs).

The Phrasebook decides, following the specification given by the interpretation, which action to take on the OpenMath object. Assume that the action is known and thus also the back-engines to be used for the computation. The Phrasebook prepares the queries for the selected back-engines by using the interpretation for distributing the computations. In simple cases, the control information corresponds directly to a single query (certain user-commands) in a single back-engine. In general however, the Phrasebook is able to extract from the control information and the OpenMath object an algorithm that splits up the problem and distributes the solving process among several back-engines. The queries sent to the back-engines consists of expressions in the back-engine syntax. These are produced in part by calling the codec's decoder on the relevant OpenMath fragments. Communication happens at the shell interface of the various back-engines because this is the easier solution for turning third-party software into a server in absence of ad-hoc solutions, e.g. most software packages do not provide a callable library of functions. The Phrasebook assembles the results it has received and produces OpenMath output from them. This output is displayed/sold to the user in a suitable form.

Phrasebooks providing an interface to and from OpenMath have been built into experimental versions of both Axiom and Gap, cf. [**?**,**?**]. We however, have adopted the approach of building the full Phrasebook outside the software package we want to interface. By use of the Naomi Java library, such Phrasebooks have already been built for the proof checkers Lego and Coq, for the computer algebra package Mathematica and for the Group Theory package Gap.

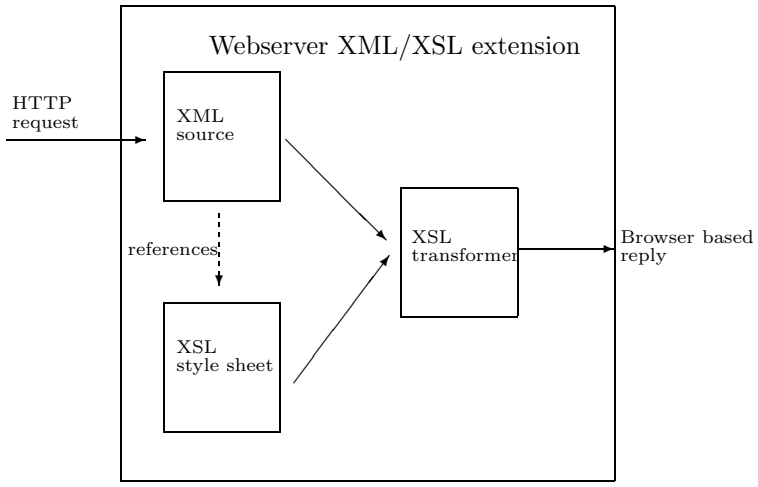
3 Browser – Webserver Architecture for an Interactive Mathematical Document

In this section we describe an architecture of an Interactive Mathematical Document based upon the OpenMath technologies as described in the previous section.



The document can be accessed by a remote client through a webserver using a webbrowser. The HTTP protocol governs the transactions between the browser and webserver as depicted in Figure **??**. In order to complete a transaction, the webserver needs to know certain information about its client, for instance the kind of browser issuing the request. This influences, among other things, the way a document is rendered.

When dealing with XML sources that contain mathematical expressions, such as OpenMath Documents, the possibility of directly accessing and controlling the webserver behaviour is very powerful **********WHY**********. In our experiments, we used an XML/XSL extension for a popular webserver [**?**] and tested our approaches with Mozilla and Netscape [1], the browsers known to the webserver extension. In general, such an extension can be visualized as in Figure **??**.



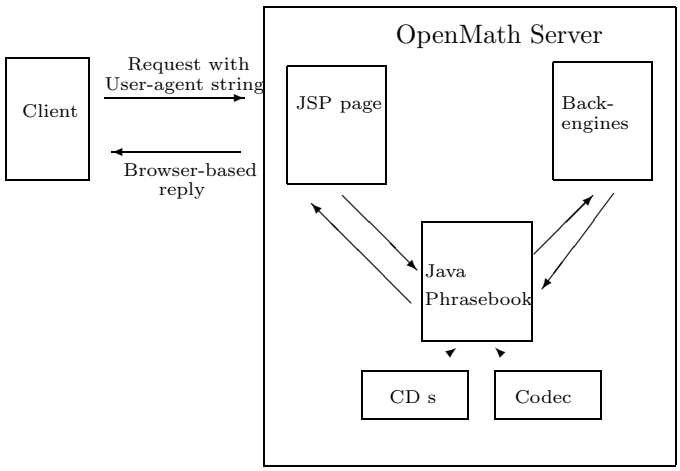
Upon receiving request for displaying an XML source, the webserver inspects the processing instructions included in the header of the source.

[1] Our work in progress is visible at `http://ruby.win.tue.nl:8080/ida`, when the server is up.

These instructions reference the XSL stylesheet(s) to be used and determine the transformation that needs to be done. The XML engine then dispatches the request to an XSL transformer and obtains the resulting transformed document which will be sent back to the client. The obvious advantage to this approach is that we need to maintain only the XML source, the various flavours of HTML that are presented to the clients do not exist on the server site but are generated upon request.

*********What is JSP***************

The OpenMath documents are linked to OpenMath servers. In particular, the OpenMath server consists of JSP pages for the interaction with the client and a Java Bean Phrasebook interpreting the request and handling the interaction with back-engines. The results of the back-engines are then, after being interpreted and handled by the Phrasebook displayed in a responce JSP page.



4 OpenMath in Algebra Interactive!

The architecture described in the previous section has been implemented for the second version of *Algebra Interactive*.

In this section we describe a few examples explaining how the OpenMath Technologies are used in this forthcoming version of *Algebra Interactive*.

4.1 Style sheets for presentations

In *Algebra Interactive* style sheets play two roles. First of all we can use XSL style sheets to obtain various kind of output formats.

In *Algebra Interactive* for example, we produce besides the interactive HTML version of *Algebra Interactive*, also a LaTeX version. This LaTeX source serves as the basis for a booklet accompanying the interactive document. In ???? you see the output of this transformation of the source file displayed in Figure ????.

*********Picture of LaTeX output********* *********Pictures of two types of output*********

Secondly, XSL style sheets are used to enhance user adaptivity to *Algebra Interactive*. Different style sheets are used to produce summaries of the chapters, a full text version, an exercise book, versions with examples and proofs adapted for various types of science students. These options are offered to the user via a menu.

The structure of the OMDOC source of *Algebra Interactive* makes it possible to feed it to a mathematical data base like MBase, [**?**]. This opens various new possibilities for creating interactive documents.

In this data base the structure of dependencies of the various items of *Algebra Interactive* are stored, thus enabling the user to automatically generate tailor-made books: a student interested in finite fields can choose to make a book covering the theory of finite fields and all its prerequisites.

4.2 Use of back engines

Even in a first year algebra course as *Algebra Interactive*, in which the basic theory of groups, rings and fields is covered, there is no computer algebra system that can do all calculations involved. In *Algebra Interactive* we make use of the systems *Mathematica*, Gap and *CoCoA* to enliven mathematics by dynamic examples and exercises. Below you find examples explaining the use of these computer algebra systems in *Algebra Interactive*.

Example 1 (Basic Arithmetic). One of the basic results in elementary number theory is the *Prime Number Theorem*:

The number of primes in the interval $[1, n]$ is of order $n/\log n$.

In a dynamic example a student can type in a number $n \in \mathbb{N}$. The back engines Gap, *CoCoA* and *Mathematica* can compute the number $prime(n)$ of primes in the interval $[1, n]$ which is then returned and displayed in the text. However, of these three packages only *Mathematica* can compute $n/\log n$ and compare it with $prime(n)$. And moreover, only *Mathematica* can graph both $prime(n)$ and $n/\log n$.

*****PLaatje******

In *Algebra Interactive* the user can specify the computer algebra system to use. In this example, *Mathematica* provides evidently more information.

Example 2 (Group and Ring Theory). Although *Mathematica* and Gap are capable of handling polynomials, *CoCoA* is better suited for dealing with subtler questions in Ring Theory. For example, to check that the ideal $(5, X^2+X+1)$ is maximaal in $\mathbb{Z}[X]$ can only be established directly by *CoCoA* but not by the other two packages.

The package Gap is certainly the best when dealing with groups. But even in Group Theory one may want to make use of the graphical

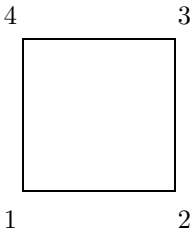
power of *Mathematica*. A finite permutation group is presented in Gap by a Schreier Tree, given as a record, no other means of vizualisation is available in Gap. With *Mathematica* we can display such a tree within a picture.

*****PLaatje******

Example 3 (Exercises). *Algebra Interactive* contains various types of exercises. There are multiple choice questions, open exercises of theoritical type and open exercises of computational type. For the latter sort the available computer algebra systems are used tot check a students answer.

An example of such an exercises is the following:

> Give generating automorphisms of the automorphism group of the square as permutations of the four vertices $1, \ldots, 4$.



Here a student can give a lot of good answers, varying from $(1, 2, 3, 4)$, $(1, 3)$ to the list of all 8 symmetries. The package Gap however, can easily check correctness.

5 Conclusions

Advantage of OpenMath and OmDoc

- Presentation in several formats, HTML, LaTeX, ...
- Presentation at different levels
- User addaptive presentation.

Advantages of the use of OpenMath servers

- Use of several back-engines for various dynamical examples
- Use of back-enigines for checking exercises
- Use of back-enigens for producing visualisations

Disadvantages

- hard to write OpenMathDoc and OpenMath.
- Various techniques are needed: Java, JSP, XML, GAP, Mathematica, OpenMath,...
- There are not enough tools available.
- No way good way for rendering mathematics