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

Abstract. New technologies such 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 communicating 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 of *Algebra Interactive*, an interactive course on first and second year university algebra.

1 Introduction

In *Algebra Interactive* "version I" [10] our purposes were to use emerging technologies of that era (the mid nineties) to bring together elements of playfullness, visualization and interactivity into teaching material in such a way that students would be exposed to the power of abstract algebra by experimenting with examples and applications closer in spirit to the ways of a working mathematician than those in classical textbooks. At that time, HTML and JAVA were the prime technological resources available to produce and enliven hypertext. In the meantime, the breakthrough of the Internet brought forward the concept of a global village of remote resources that can also be used for doing mathematics. To bring mathematics to the Internet, new markup languages and related extensions such as XML, XSL and both MATHML and *OpenMath* are already in an advanced stage of development and are gaining ground rapidly. The experience with *Algebra Interactive* I on the one hand and the new technologies evolving rapidly on the other hand, has helped shape the ideas for this paper and for the forthcoming second version of *Algebra Interactive*. In rough terms we are approaching the stage where 'What you want is what you get'[1] is the philosophy underlying the interactive document. How we as authors envision to achieve this goal as best as possible is the topic of this paper. Specifically, the 'you' in the above quote refers to tutors and students. To satisfy the requirements of these user groups, the electronic document has to take into account flexibility of presentation, easy access of back-engines, indexing, search and test

[1] We first heard this from Gaston Gonnet.

facilities. All of this can be accomplished by exploiting *OpenMath* and OmDoc in combination with server side programming.

The structure of this paper is as follows. Section 2 discusses *OpenMath*, MathML and related technologies. Section 3 is concerned with the browser-webserver architecture. Finally, we describe the application of these ideas in *Algebra Interactive* in Section 4.

2 *OpenMath*, MathML and related technologies

A lot of the mathematics online has until recently been presented using the de-facto standard LaTeX or TeX despite the well known limitations it has in handling interactivity. Documents generated by transforming TeX sources to various flavors of html support only cross-linking mechanisms and produce, in most cases, images for formulae. More adventurous authors have included interactivity in the form of plugins or Java applets in their pages. In the meantime, the World Wide Web Consortium has published the recommandation for the Mathematical Markup Language (MathML) for describing mathematical notation and capturing both its structure and content [2]. Similarly, the *OpenMath* Society has endorsed a newer version of the *OpenMath* standard for representing the semantics of mathematical objects and facilitating the exchange between computer programs, the storage in knowledge bases, and the electronic publication of mathematics [15,11,9].

2.1 *OpenMath* and MathML

By enabling mathematics to be served, received, and processed electronically, both MathML and *OpenMath* aim at enhancing mathematical communication on the World Wide Web just as hypertext has done for text. The parallel evolution and definition of these predominant markup languages for mathematics [7] resulted in an ideal separation of complementary rôles: MathML presentation can be used for presenting mathematical content written in *OpenMath*. The major observation underlying this state of affairs is that *mathematics and its presentation should not be viewed as one and the same thing.* While the meaning of a mathematical object should be uniquely defined and understood, its visualization and rendering depends on time and place, more precisely it depends on the context and on the style preferences of the author or reader.

In order to represent mathematical content it is crucial that a mechanism is established to allow for the introduction of new concepts, since this activity is at the core of mathematics. The *OpenMath* representation of mathematics relies for this task on a small set of "expression trees" constructors (application, binding, attribution and error), on some basic objects (bytearrays, strings, integers, IEEE floats, variables), and on the usage of symbols defined in Content Dictionaries (CDs for short). These are publicly available collections of mathematical definitions, a sort of xml dictionary of mathematics. The interested reader may find the standard documents and the collection of public Content Dictionaries from [20].

OpenMath abstract objects representing mathematical concepts are obtained for instance by using the **application** constructor as in the following example. The algebraic structure consisting of the polynomial ring $\mathbb{Z}_p[X]$ is representable as *OpenMath* abstract object like:

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

In this object, the *OpenMath* symbols `polyr:PolynomialRingR` and `setname2:Zm` identify the polynomial ring structure over the integers modulo p. More precisely, they are the symbols called `PolynomialRingR` and `Zm` defined in the CDs `polyr` and `setname2`, respectively. As mentioned earlier, *OpenMath Content Dictionaries* collect and provide definitions of mathematical notions for usage within *OpenMath* applications. The official repository for CDs is [20].

As further example consider a polynomial in this ring, say $f = X^3 - X + 1$. It can be represented in several ways as an abstract *OpenMath* object, for instance as recursive polynomials by using the symbol `polyr:PolynomialR`. As with all *OpenMath* objects, it can be encoded in a human-readable format using XML [14] 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>
```

Reading from the encoding, 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).

While *OpenMath* objects are built using symbols defined in some Content Dictionary that is part of an ever growing collection of Content Dictionaries, MathML makes explicit a relatively small number of commonplace mathematical constructs chosen within the K-12 realm of applications and, in addition, it provides the content symbol (`csymbol`) to introduce a new symbol whose semantics is not part of the core content elements of MathML. In particular, such an external definition may reside in an *OpenMath* Content Dictionary as in the example:

```
<apply>
   <csymbol encoding="OpenMath"
           definitionURL="http://www.openmath.org/cd/polyr.ocd">
           PolynomialR
   </csymbol>
   <apply>
       <csymbol encoding="OpenMath"
               definitionURL="http://www.openmath.org/cd/setname2.ocd">
               Zm
       </csymbol>
       <ci>p</ci>
   </apply>
   <ci>X</ci>
</apply>
```

MathML supports an extensive library of presentation symbols to accomodate even the fanciest notation used by mathematicians. In this respect, MathML (presentation) is the preferred choice for rendering mathematical content and as such it is to be expected that it will be natively understood by future browsers (e.g. Mozilla already ships with MathML) and editors.

2.2 *OpenMath* and *OpenMath* Documents

Communication of mathematics is however more involved than just an exchange of stand-alone mathematical notions or objects. Often it is crucial to be able to relate concepts, to link definitions to theorems and in general to convey an entire mathematical theory. In principle, one could use *OpenMath* (attribution) objects to represent this type of information, in a way similar to the way in which Content Dictionaries can be expressed as *OpenMath* objects. However, the resulting representation would be quite unnatural and cumbersome to say the least. The *OpenMath* Document Specification (OmDoc) [18], currently under development, is an XML document type definition that can be used to represent general mathematical knowledge as it appears 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* [10], an interactive textbook used in teaching first and second year university algebra, see Section 4. *OpenMath* Documents are moreover intended as the input format for MBase [19], a knowledge base of mathematics.

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 marked 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.

Figure 1 contains a fragment of an *OpenMath* Document that defines in the natural way the binary predicate divides for natural numbers.

```
<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>
...
```

Fig. 1. OmDoc source fragment.

For generality, the mathematics in *OpenMath* Documents is represented using *OpenMath* objects. Although *OpenMath* Documents are readable, one cannot exactly describe them as user-friendly or appealing. High-quality presentation of the content in OmDoc format can be generated using XSL stylesheets. See for instance Fugure 5. Manipulations on the source XML documents can be conveniently and transparently performed in a browser-webserver architecture.

3 Browser – Webserver Architecture for Interactive Mathematical Documents

In this section we describe a browser-webserver based architecture used for supporting the interactivity we desire in mathematical documents. This architecture exploits the *OpenMath* technologies described in the previous section.

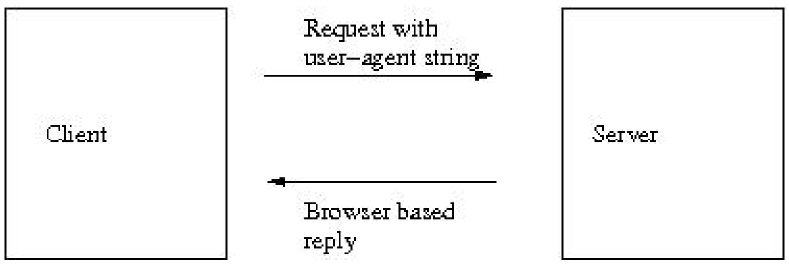


Fig. 2. General client-server contract.

All the mathematical documents we are concerned with can be accessed through a webserver by a remote client, for instance a browser. This implies that the HTTP protocol which governs the transactions between browsers and webservers as depicted in Figure 2 is ultimately responsible for fetching and presenting the document. In order to complete such 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 on the screen.

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. For instance it is possible to trigger a specific transformation on the source document depending on the geographic location of the requesting client or on the specific browser used to view the document. Mathematical concepts can be accompanied by a translation in the suitable language or, more importantly, the notation used for the rendering of expressions can be tuned to the regional usage (the interval $0 < x \leq 1$ would be written $(0, 1]$ by an Anglo-saxon, but $]0, 1]$ by, say, a Frenchman). Different mechanisms of presentation can be invoked for different browsers: a reply containing MathML can be returned to MathML-enabled browsers, whereas a reply containing images or applets can be returned to simpler browsers.

In our experiments, we used an XML/XSL extension for a popular webserver [1] and tested our approaches with Mozilla and Netscape [2], the

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

browsers known to the webserver extension. In general, such an extension can be visualized as in Figure 3.

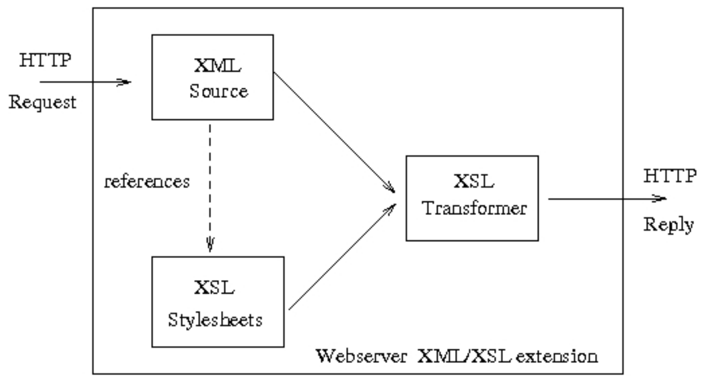


Fig. 3. XML-XSL webserver extension..

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

Additionally, server programming comes into play when interaction with an *OpenMath*server takes place. *OpenMath* servers are servers that interface via *OpenMath* to back-engines. They are used to support the computational requests issued interactively from the document.

In particular, the *OpenMath* servers used in *Algebra Interactive* perform two tasks:

– interpret the request coming from the client and handle the interaction with the back-engines,
– display the results and the graphical user interface using JAVA server pages.

3.1 *OpenMath* Servers

An *OpenMath* server is a server consiting of a combination of (possibly one) mathematical back-engines interfaced by *OpenMath* Phrasebooks. Such servers handle requests of a mathematical nature. Some examples are given in Section 4.

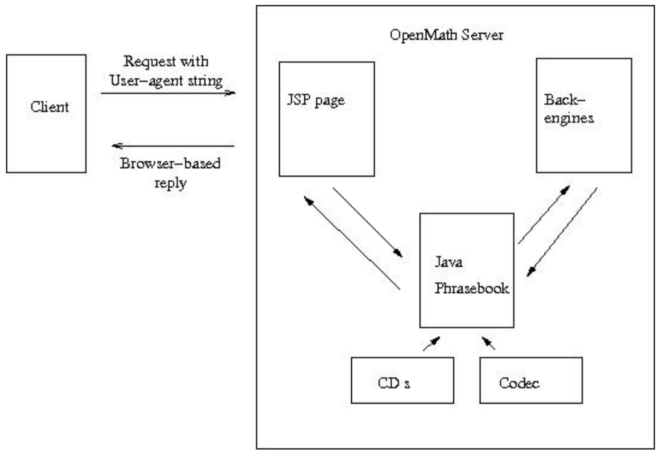


Fig. 4. *OpenMath* server for *Algebra Interactive*.

An *OpenMath* Phrasebook is a program able to handle *OpenMath* objects provided that they are built using the CDs that the Phrasebook recognizes. A Phrasebook is able to perform several tasks, like for instance evaluation, simplification, proving, solving, and printing. A Phrasebook also specifies how the actual *communication* between the software package and the *OpenMath* computer environment is achieved.

Upon receiving the request, the Phrasebook decides which action to take on the *OpenMath* objects and prepares the queries for the selected back-engines, possibly by distributing the computations across a network [4]. In simple cases, the request corresponds directly to certain user-commands for a single back-engine. In the case in which the back-engine is *OpenMath*-aware and able to handle *OpenMath* directly, the Phrasebook sends the objects along with the queries without further intervention. However, it is also possible to take advantage of software which is not *OpenMath*-aware by adding a translation layer into the Phrasebook directly [5]. In this last case, the queries sent to the back-engines consist of expressions in the back-engine syntax produced by the Phrasebook on the relevant *OpenMath* fragments. At last, the Phrasebook assembles the results it receives and produces *OpenMath* output from them. This output is then presented 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. [12,13]. Other computer algebra packages include modules able to handle *OpenMath* directly [3,17]. A possible approach of building the full Phrasebook outside the software package is described in [5]. By use of the JAVA libraries for *OpenMath*, such external Phrasebooks have been implemented for the proof checkers Lego and Coq, for the computer algebra packages Maple, *Mathematica* and Gap [8,6].

4 *OpenMath* in *Algebra Interactive*

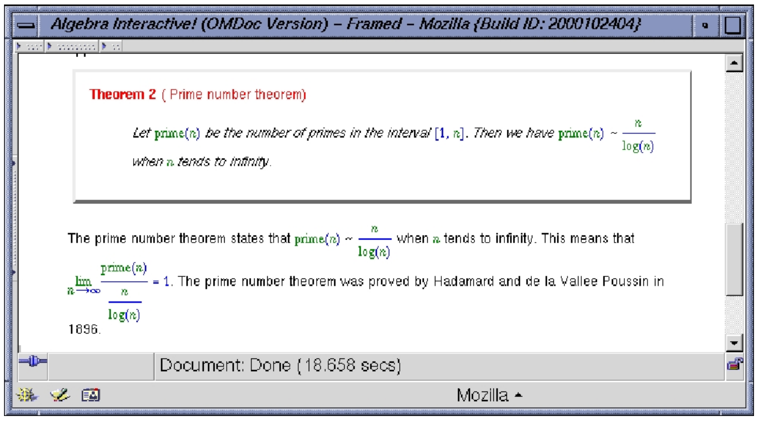
The architecture described above has been implemented for the second version of *Algebra Interactive*, thus enriching the technology used in the first version of *Algebra Interactive*. In this section we describe a few examples explaining how the *OpenMath* technologies are used in the forthcoming version of *Algebra Interactive*.

4.1 Presentations of *Algebra Interactive*

The source files of *Algebra Interactive* are OMDOC files. As explained before these sources can be made presentable by translating them into other formats by invoking XSL transformations.

In *Algebra Interactive* we use XSL stylesheets to enhance user adaptivity. In particular, various stylesheets are used to produce, for example, a full text version of the document, summaries of the sections and chapters, an exercise book, and versions with examples and proofs adapted for various types of science students. At the moment, the text of *Algebra Interactive* is in English, but we envision also other languages (e.g., Dutch or German) to be available in the future.

In the interactive version of *Algebra Interactive* the options are offered to the user via a menu. Besides the interactive HTML versions of *Algebra Interactive*, XSL stylesheets are also used to produce LaTeX versions. [3] In Figure 5 one sees the output, rendered by Mozilla, of a transformation of the source file displayed in Figure 1 into an HTML file containing an overview of the material covered.


Algebra Interactive! (OMDoc Version) – Framed – Mozilla {Build ID: 2000102404}

Theorem 2 (Prime number theorem)

Let $\mathrm{prime}(n)$ *be the number of primes in the interval* $[1, n]$. *Then we have* $\mathrm{prime}(n) \sim \frac{n}{\log(n)}$ *when* n *tends to infinity.*

The prime number theorem states that $\mathrm{prime}(n) \sim \frac{n}{\log(n)}$ when n tends to infinity. This means that $\lim_{n\to\infty} \frac{\mathrm{prime}(n)}{\frac{n}{\log(n)}} = 1$. The prime number theorem was proved by Hadamard and de la Vallee Poussin in 1896.

Document: Done (18.658 secs)

Fig. 5. Detail of one page of *Algebra Interactive* as rendered by Mozilla.

[3] These LaTeX sources can serve as the basis for booklets accompanying the interactive document.

The structure of the OmDoc source of *Algebra Interactive* makes it possible to feed it to a mathematical data base like MBase, [19]. Here we envision various new possibilities for creating interactive documents. For, in this data base the structure of dependencies of the various items of *Algebra Interactive* can be 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. This line of research is presently being pursued by the Omega group at Saarbrücken University and Riaca at Eindhoven University of Technology.

4.2 Use of computational back-engines

Even in a basic algebra course such as *Algebra Interactive*, in which groups, rings and fields are treated, there is no single computer algebra system suitable for directly handling all calculations involved. In *Algebra Interactive* we make use of the systems *Mathematica* [21], Gap [16] and *CoCoA* [3] to enliven mathematics by dynamic examples and exercises. Below you find examples explaining the use of these computer algebra systems in the online version of *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 $n \in \mathbb{N}$. All three back-engines Gap, *CoCoA* and *Mathematica* are able to compute the number prime(n) of primes in the interval $[1, n]$; the result is then returned and displayed in the text. However, of these three packages only *Mathematica* is able to compute $n/\log n$ and compare it with prime(n). Moreover, only *Mathematica* can graph both prime(n) and $n/\log n$.

In *Algebra Interactive* the user can specify the computer algebra system to use. In this example, *Mathematica* provides evidently more information. [4]

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, checking maximality of the ideal $(5, X^2 + X + 1)$ 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*.

To each permutation group G on a set Ω, say, we can associate a graph Γ with vertex set Ω and edge set an orbit of G on the pairs from

[4] A user without access to *Mathematica* can always use the free software packages Gap and *CoCoA*.

Ω. The group G is contained in the automorphism group of Γ. With Gap the edge set can easily be determined but in Gap no means of graphical visualization is available. With *Mathematica* we can display the graph within a picture.

As an example, consider the group

$$G = \langle (2,5)(3,6)(4,7), (1,5,2)(3,6,8)(4,7,9), (1,5,8,10,4)(2,6,9,3,7) \rangle,$$

acting on the set $\Omega = \{1,\dots,10\}$. The G-orbit E of the pair $\{1,8\}$ has length 15. The corresponding graph $\Gamma = (\Omega, E)$ is the Petersen graph.

Below you see the Gap computation and *Mathematica*'s graphical presentation.

```
gap> G:=Group((2,5)(3,6)(4,7),(1,5,2)(3,6,8)(4,7,9),(1,5,8,10,4)(2,6,9,3,7));
Group((2,5)(3,6)(4,7),(1,5,2)(3,6,8)(4,7,9),(1,5,8,10,4)(2,6,9,3,7))
gap> Orbit(G,[1,8],OnSets);
[ [ 1, 8 ], [ 3, 5 ], [ 5, 10 ], [ 2, 6 ], [ 7, 8 ], [ 2, 10 ], [ 4, 8 ],
  [ 6, 9 ], [ 3, 9 ], [ 1, 10 ], [ 4, 6 ], [ 3, 7 ], [ 4, 5 ], [ 1, 9 ],
  [ 2, 7 ] ]
```

```
<<DiscreteMath`Combinatorica`

In[1]:= ShowGraph[MakeGraph[{1,2,3,4,5,6,7,8,9,10},
     MemberQ[{{1,8},{3,5},{5,10},{2,6},{7,8},{2,10},{4,8},
     {6,9},{3,9},{1,10},{4,6},{3,7},{4,5},{1,9},{2,7}},
     {#1,#2}&]]]
```

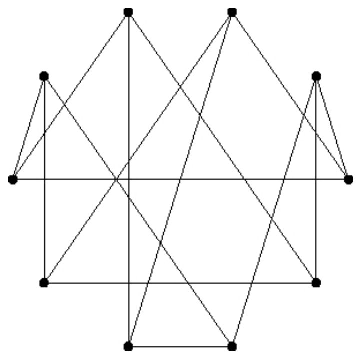


Fig. 6. *Mathematica*'s Petersen graph.

The communication between the two packages is established using an *OpenMath* server. The Gap output

```
([1..10],[ [ 1, 8 ], [ 3, 5 ], [ 5, 10 ], [ 2, 6 ],
   [ 7, 8 ], [ 2, 10 ], [ 4, 8 ], [ 6, 9 ], [ 3, 9 ], [ 1, 10 ],
   [ 4, 6 ], [ 3, 7 ], [ 4, 5 ], [ 1, 9 ], [ 2, 7 ] ]
```

is translated into the following *OpenMath* syntacs for the graph Γ, which consits of a (vertex) set and an edge set:

```
<OMOBJ>
 <OMA>
   <OMS cd="ida" name="graph"/>
   <OMA>
      <OMS cd="set1" name="set">
      <OMV name="1" id="1"/> ... <OMV name="10" id="10"/>
   </OMA>
   <OMA>
      <OMS cd="set1" name="set">
      <OMA>
         <OMS cd="ida" name="edge"/>
         <OMV name="1" xref="1"/>
         <OMV name="8" xref="8"/>
      </OMA>

      ...

       <OMA>
         <OMS cd="ida" name="edge"/>
         <OMV name="2" xref="2"/>
         <OMV name="7" xref="7"/>
      </OMA>
   </OMA>
 </OMA>
</OMOBJ>
```

The *Mathematica* phrasebook takes care of the translation into *Mathematica* syntax

```
MakeGraph[{1,2,3,4,5,6,7,8,9,10},
      MemberQ[{{1,8},{3,5},{5,10},{2,6},{7,8},{2,10},{4,8},
     {6,9},{3,9},{1,10},{4,6},{3,7},{4,5},{1,9},{2,7}},
     {#1,#2}&]]
```

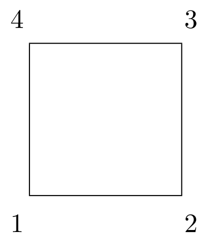
and sends it to *Mathematica* for producing the graphical presentation.

Sometimes naturally occurring questions in basic algebra require the usage of more than one back-engine. Consider the ring $R = \mathbb{Z}_p[X]/(f)$, where $\mathbb{Z}_p$ denotes the field with p elements, p prime, and f is a polynomial of positive degree in $\mathbb{Z}_p[X]$. If $a \in R$ is an invertible element, then multiplication by a determines a permutation $\sigma_a : R \to R$ of the finite set R. Group theory aspects of such permutations are best studied in Gap, whereas *CoCoA* is best suited for studying the ring theoretic aspects. For example, questions related to the order of σ_a or type of group generated by permutations of the form σ_b, with b invertible, are best solved by Gap. On the other hand, finding invertible elements in R giving rise to a specific permutation requires a computation better suited for *CoCoA*.

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

An example of such an exercise 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 many good answers, varying from the rotation $(1, 2, 3, 4)$ and reflection $(1, 3)$ to the list of all 8 symmetries of the square. Instead of listing all possible answers, we use the computing power of Gap to check correctness. Moreover, by using Gap it is even possible to spot certain types of errors in wrong answers. For example, if the permutations given in an answer do not generate the full group, the student is prompted to add more automorphisms. If a permutation is not an automorphism, then the student is asked to remove it from the answer and think of other generators.

Similarly, questions concerning generators of ideals can have infinitely many correct answers. Answers to such questions are best checked with a back engine, e.g. *CoCoA*.

5 Conclusions

In this paper we discussed some new *OpenMath* and OmDoc technologies and their applications to interactive mathematical documents. In particular, we described the new features that are being incorporated in the forthcoming second version of *Algebra Interactive*.

Advantages of using *OpenMath* and OmDoc in mathematical documents are:

- Documents can be presented in several formats, like HTML, XHTML, LaTeX, pdf, PostScript, ...
- The presentation of the content is flexible and can be adapted to the user. From the same source one can make a full text version as well as summaries or sheets for presenting the material in class.
- The use of several computational back-engines makes it possible to create many dynamical examples, including visualization of mathematical concepts and automated theorem proving.
- Computer algebra systems can be used for checking exercises.

Besides these advantages we should also mention some drawbacks. In the approach we have taken, the source documents have been marked up using the OmDoc format. Producing such sources is a tedious and time consuming job, as, at present, there are no good authoring tools available.

Also the set up of an *OpenMath* server with several computer algebra systems as back-engines is cumbersome, as a developer needs to master various technologies, such as *OpenMath*, JAVA, JSP, Gap, and *Mathematica*.

References

1. Apache XML Project. `http://xml.apache.org/cocoon`, June 2000.
2. Ron Ausbrooks, Stephen Buswell, Stphane Dalmas, Stan Devitt, Angel Diaz, Roger Hunter, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) Version 2.0. W3C Working Draft 28 March 2000, March 2000. Available at `http://www.w3.org/TR/REC-MathML2/`.
3. A. Capani, G. Niesi, and L. Robbiano. *CoCoA, a system for doing Computations in Commutative Algebra*, 1995. Available via anonymous ftp from `cocoa.dima.unige.it`.
4. O. Caprotti, A. M. Cohen, H. Cuypers, M. Riem, and H. Sterk. Using OpenMath Servers for Distributing Mathematical Computations. In 5^{th} *Asian Technology Conference in Mathematics*, Chiang-Mai, Thailand, December 17–21 2000.
5. O. Caprotti, A. M. Cohen, and M. Riem. JAVA Phrasebooks for Computer Algebra and Automated Deduction. *SIGSAM Bulletin*, 2000. Special Issue on OpenMath.
6. O. Caprotti and M. Oostdijk. How to formally and efficiently prove prime(2999). In *Proceedings of Calculemus 2000: 8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning*, St. Andrews, Scotland, August 2000.
7. Olga Caprotti and David Carlisle. OpenMath and MathML: Semantic Mark Up for Mathematics. *Crossroad*, Special Issue on Markup Languages, 1999. `http://www.acm.org/crossroads`.
8. Olga Caprotti and Arjeh Cohen. Connecting proof checkers and computer algebra using OpenMath. In *The 12th International Conference on Theorem Proving in Higher Order Logics*, Nice, France, September 1999.
9. Olga Caprotti and Arjeh Cohen. On the role of OpenMath in interactive mathematical documents. *Journal of Symbolic Computation*, Special Issue on the Integration of Computer algebra and Deduction Systems, 2000.
10. A. M. Cohen, H. Cuypers, and H. Sterk. *Algebra Interactive!* Number ISBN 3-540-65368-6. Springer Verlag, 1999.
11. OpenMath Consortium. The OpenMath Standard, August 1999. O. Caprotti, D. P. Carlisle and A. M. Cohen Eds.
12. OpenMath Consortium. Axiom interface to OpenMath. OpenMath ESPRIT Deliverable, 2000.
13. OpenMath Consortium. GAP interface to OpenMath. OpenMath ESPRIT Deliverable, 2000.
14. World Wide Web Consortium. Extensible Markup Language (XML) 1.0. W3C Recommendation REC-xml-19980210, February 1998. Available at `http://www.w3.org/TR/1998/REC-xml-19980210`.
15. S. Dalmas, M. Gaëtano, and S. Watt. An OpenMath 1.0 Implementation. pages 241–248. ACM Press, 1997.
16. The GAP Group, Aachen, St Andrews. *GAP – Groups, Algorithms, and Programming, Version 4.2*, 1999. `http://www-gap.dcs.st-and.ac.uk/~gap`.

17. Anthony C. Hearn. *User's Manual: Version 3.6.* RAND Publication CP78 (Rev. 7/95), Santa Monica, CA 90407-2138, July 1995.
18. M. Kohlhase. OMDoc: Towards an Internet Standard for the Administration, Distribution and Teaching of mathematical Knowledge. In *Proceedings of AISC '2000, Artificial Intelligence and Symbolic Computation, Theory, Implementations and Applications*, Madrid, Spain, July 2000.
19. Michael Kohlhase and Andreas Franke. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation*, 2000. Special Issue on the Integration of Computer algebra and Deduction Systems.
20. OpenMath Society Website. `www.openmath.org`.
21. Stephen Wolfram. *The Mathematica Book: Fourth Edition.* Cambridge University Press, 1999. `http://documents.wolfram.com/framesv4/frames.html`.