Interactive Mathematical Documents on the Web

Arjeh M. Cohen, Hans Cuypers, Ernesto Reinaldo Barreiro, and Hans Sterk

Department of Mathematics,
Technische Universiteit Eindhoven,
POB 513,
5600 MB Eindhoven,
The Netherlands

Abstract. This paper deals with our work on interactive mathematical documents. These documents accomodate various sources, users, and mathematical services. Communication of mathematics between these entities is based on the OpenMath standard and Java technology. But, for the management of the communication, more protocols and tools are needed. We describe an architecture that serves as a framework for our work on interactive documents, and we report on what we have implemented so far.

1 Introduction

An interactive mathematical document is to be regarded as a book, an article, or announcement on computer that can be read in the way their ordinary (paper) counterparts can, but which, in addition, enables a variety of activities. Among these interactions, we count storing and communicating mathematics, presenting mathematics (e.g., in browsers) and performing mathematical operations, possibly elsewhere on the Web.

Although the notion of an interactive mathematical document has been around for several years [14] its realization is nowhere near the final stage. Recent web technological progress, for instance, has enabled a smoother communication of mathematics than ever before. The use of an interactive mathematical document can provide a window to the world of mathematical services on the internet. Moreover, a mathematical service on the internet can be created by the construction of an interactive mathematical document. In §3 of this paper we paint the contours of such a mathematical document. In particular, we describe a *mathematical document server* which takes input from various sources (the document source, mathematical services, and users), creates a highly interactive mathematical set-up, and serves it to the user. In this vein, the paper can be viewed as a sequel to [11].

We would like to stress that the technology for achieving such goals already exists, mainly (for our purposes) in the form of Java software. In §4, we describe some of the main tools we have developed for realizing the interactive mathematical documents. We envision that it will require a few

more years before integrated authoring tools, as easy to use as LaTeX, will be widely available. With the work presented here, we intend to contribute to these developments.

Before going into details regarding interactive documents, in §2, we discuss the general picture of mathematics on the Web.

2 A Framework for Interactive Mathematics

In this section we describe our approach to interactive mathematics. In §2.1, we begin by overviewing OpenMath, a standard for communicating mathematics across the Web. Next, in §2.2, we discuss web services and, in particular, address two additional requirements for our purposes: query facilities and a notion of the state of the mathematics that is being communicated.

2.1 OpenMath

A starting point for semantically rich communication across the Web is the standard for mathematical expressions OpenMath, cf. [25], and, for our purposes, its XML encoding. The representation of mathematics in OpenMath relies on four 'expression tree' constructors (viz., application, binding, attribution, and error), on five basic objects (byte arrays, strings, integers, IEEE floats, variables), and on a special, sixth, basic object: symbols, defined in Content Dictionaries (CDs for short). The core CDs are publicly available collections of mathematical definitions. The standard documents and the collection of public CDs for OpenMath are available in XML format from [25]. An example of the XML encoding of an OpenMath object expressing that $\cos(\pi) = -1$ is given in Figure 1.

```
<OMOBJ>
 <OMA>
  <OMS cd="relation1" name="eq"/>
  <OMA> <OMS cd="transc1" name="cos"/> <OMS cd="nums" name="pi"/> </OMA>
  <OMI>-1</OMI>
 </OMA>
</OMOBJ>
```

Fig. 1. OpenMath fragment.

For a further introduction to OpenMath, see [11]. There it is also explained how OpenMath and MathML [21] complement each other in that OpenMath objects express mathematical content whereas MathML mainly focuses on presentation. The connection between the two rests upon

- the fact that MathML works with a relatively small number of commonplace mathematical constructs chosen within the high school realm of applications and

- the content symbol (`csymbol`) in MathML for introducing a new symbol whose semantics is not one of the core content elements of MathML. In particular, such an external definition may reside in an OpenMath Content Dictionary.

For purposes of alignment of MathML and OpenMath, the core CDs contain symbols matching the MathML constructs.

As it stands, the OpenMath mechanism works quite well for conveying mathematical objects: by declaring which CDs are relevant, two parties agree on a common understanding of the mathematics they communicate. The public CDs are (well-wrought) examples, but two parties may choose whichever CDs they like. They could even create CDs for the sole purpose of a brief communication. This feature ensures a great flexibility in the use of OpenMath.

In Figure 2, by way of example, we display an experimental CD for planar Euclidean geometry, which was recently constructed in joint work with Ulrich Kortenkamp for the purpose of interfacing with Cinderella, [29].

Phrasebooks provide the means to convert OpenMath objects to/from software applications. They parse OpenMath objects into an application-native language (e.g., *Mathematica*, Maple, GAP), sending the result to the application, catching the response from the application, and translating it back into OpenMath. The phrasebook communicates only OpenMath objects of which the symbols are defined in the CDs that the phrasebook recognizes. Thus, a phrasebook performs the translation back and forth as well as the communication. The actual task performed by the totality of the phrasebook actions depends on the interpretation. If the application is a computer algebra system, the interpretation is often 'evaluation' or 'simplification': when passed $2+3$, these applications will return 5. If the application is a proof assistant (e.g., Lego or Coq, cf. [9]), then 'verifying' or 'proving' is a more likely interpretation of what the application is supposed to do, and if the application is a browser or printing, the interpretation is to prepare the mathematical object for a presentation.

Phrasebooks providing interfaces to and from OpenMath have been built into Axiom and GAP [2,16].

We have developed a Java library, called ROML, for building full phrasebooks outside mathematical software packages. It is described in [4] and can be found at [28]. By use of ROML, such external phrasebooks have been implemented for the proof checkers Lego and Coq, for the computer algebra packages Maple, *Mathematica*, and GAP [5,6].

2.2 Mathematical Web Services

An important mode of communicating mathematics across the Web, is by means of queries. Generally a query in Web technology refers to a request for a service, see [37]. Naturally, we would like a standard way of expressing queries, a management system for parameters accompanying the question,

```
<CD>
<CDName> plangeo1 </CDName>

 (... further data like URL, creation date, CDs on which this one depends ...)

<Description>
This CD defines symbols for planar Euclidean geometry.
</Description>

<CDDefinition>
<Name> point </Name>
<Description>
The symbol is used to indicate a point of  planar Euclidean geometry
by a variable. The point may (but need not) be subject to constraints.
</Description>
</CDDefinition>

 (... a similar definition for ‘line’ ...)

<CDDefinition>
<Name> incident </Name>
<Description>
The symbol represents the logical incidence fucntion which is a
binary function taking arguments representing
gemetric objects like points and lines and returning a boolean value.
It is true if and only if the first argument is incident to the second.
</Description>

<Example> The line l through (points) A and B is given by:

<OMOBJ>
  <OMA>
    <OMS cd="plangeo1" name="line"/>
    <OMV name="l"/>
    <OMA>
      <OMS cd="plangeo1" name="incident"/>
      <OMV name="A"/>
      <OMV name="l"/>
    </OMA>
    <OMA>
      <OMS cd="plangeo1" name="incident"/>
      <OMV name="B"/>
      <OMV name="l"/>
    </OMA>
  </OMA>
</OMOBJ>
</Example>
</CDDefinition>
</CD>

 (... further definitions ...)
```

Fig. 2. CD fragment.

and a reference mechanism to couple a message to the query which it answers. Such systems are under construction (e.g., work of Caprotti), often based on more general standards, such as the Web Service Description Language, WSDL. We refer to [32] for a discussion of computational and other Web services. Confidentiality and privacy are important user requirements that will have to be present in user profiles. Clearly, there is a need for the development of service management frameworks with adequate provision for resilience, persistence, security, confidentiality and end user privacy. Here, however, our focus is on the mathematical aspects. We shall depart from the OpenMath set-up for the communication of mathematical objects. A mathematical query usually refers to mathematical objects, which can be phrased in OpenMath, but often the user wants to convey more than just the mathematical objects themselves. As we have seen above, a mathematical object can often be interpreted as a query by a phrasebook (e.g., interpret $\cos(\pi)$ as 'evaluate $\cos(\pi)$' or interpret an assertion GRH as 'verify GRH'), but this is a poor way of formulating a query. A more elaborate mechanism is needed.

Regarding mathematical services across the Web, we face three issues that need further exploration:

- mathematical reliability,
- expressing mathematical queries, and
- taking into account the state, or context, in which a mathematical query takes place.

Reliability. In [9], the reliability (quality guarantee) aspects are emphasized. Up till now, complexity, the (estimated) time a computation will take, has been one of the major concerns regarding mathematical computations. Although it will remain useful for clients to be aware of feasibility, they will be more concerned with the validity of the answer. Here an interesting shift in focus from complexity to convincibility may take place. To gain experience with this issue, we work out (within our MathBook technology, see §3) some concrete examples where besides the usual invocation of an algorithm, additional work is carried out to provide the users with witnesses as to the truth of the answer. In one of these examples, in response to a query for the stabilizer H of the vertex, 1 say, of a permutation group G specified by generating permutations $a_1, \ldots, a_t$, one usually expects just a set $b_1, \ldots, b_s$ of permutations fixing 1, which will generate H. Now it is a straightforward check that $b_1, \ldots, b_s$ fix 1, but it requires some work to see that these permutations actually belong to G, whence to H. Expressions of the b_j as products of $a_1, \ldots, a_t$ will solve this. Finally, a proof that each element of H lies in the subgroup of G generated by $b_1, \ldots, b_s$ requires further information, corresponding to Schreier's lemma [13]. By means of a little programming at the back engine, the additional information can be supplied, and embedded in a proof in words that supplies a substantiated answer to the query.

Queries. We have argued that, so far, the OpenMath set-up seems rather primitive in that only mathematical expressions are passed, with no indica-

tion of the required action on the object. Currently, the phrasebook makes this interpretation, and so the matter is resolved by a declaration from the phrasebook of what its action (interpretation) is, see [11]. For instance, an OpenMath object like

```
Factors(Polynomial(X,X^2-1,Rationals))
```

will result in a response of the form

```
List(Polynomial(X,X-1,Rationals),Polynomial(X,X+1,Rationals)))
```

when sent to GAP, because its phrasebook tends to interpret the OpenMath object as an evaluation command, whereas the same expression would just be printed as something like "Factors of $X^2 - 1$" when sent to a typesetting program.

In [31], a mode of interaction is implemented where the behavior of a computer algebra system can be controlled from within a JSP page (see §4.2 below) by using a set of primitives such as assigning and retrieving OpenMath objects to CAS variables, manipulating variables using the language of the CAS. Indeed, this seems to come closer to the intended user control.

But, as hinted at in [9], there are probably better solutions from automated proof checking. A slight extension of the language in which we formulate mathematical assertions will enable us to formulate mathematical queries. Typed λ calculus expressions like $\Gamma \vdash ? : P$, where Γ represents the context (see below) and $? : P$ stands for the request for a proof of assertion P, are expected to embed into a full type checking mechanism without problems. In other words, we expect that it is possible to set up a language of well-formed query expressions, recognizable by means of a proper type inference algorithm. So, importing this language within the OpenMath framework, we expect to obtain a sound method of expressing queries by means of a CD defining the primitive symbols (corresponding to question marks used such as in $? : P$) for the most fundamental types of question asked. This approach to queries is as yet unexplored, and we intend to explore it in the near future.

Context. The problem of how to handle the state in which a mathematical query takes place has not been addressed in some of the more successful mathematical services on the internet, such as Sloane [30], Faugère's Gröbner basis service [15], Wilson's Atlas of representations of finite simple groups [36], Brouwer's coding theory data base [3], and Web*Mathematica* [35].

For example, Web*Mathematica* is a way of accessing *Mathematica* via the Web. Via browser pages users can formulate either full *Mathematica* commands or input for pre-programmed *Mathematica* commands (so that no specific knowledge of *Mathematica* is required) which will then be carried out by a *Mathematica* program run by a server accessible to the user. However, after the command is carried out, the *Mathematica* session is 'cleaned' in that the user can no longer refer to the previous command. So, it is possible to, say, compute the determinant of a matrix but the user cannot assign the

matrix to a variable, say A, and change an entry of A, and/or ask for A^{-1} without re-entering the entries of A.

Clearly, as is the case for a *Mathematica* session per se, it is desirable to be able to refer to the variables at hand in a work session, to be able to ask for a second computation regarding an object passed on earlier, and so on. From a computer algebra point of view, the *context* is a list of definitions, that is, assignments to variables, of objects introduced (and computed) before (think of the assignment statements and of the In[n] and Out[n] variables for $n = 1, 2, \ldots$ in *Mathematica*). In the Javamath API, discussed in [31], there is a notion of session, in which it is possible to retain variables and their values.

However, we wish to incorporate one more feature in our notion of context. This is taken from logic, where the notion of a context is our inspiration. Indeed the symbol Γ above stands for context. Besides definitions, it contains statements which are interpreted as 'the truth' (or axioms, for that matter). This means that theorems, lemmas, conjectures, and so on, may be thrown in, and are all interpreted as 'facts'. We stress that there may very well be assumptions in the context, so that it might be possible to derive a contradiction. It would not be desirable to have the starting context of an interactive book be self-contradictory, but, in the course of a user developed proof by contradiction, there is nothing against a set of assumptions from which the user can derive $0 = 1$.

It is such a list of definitions of objects and statements, which is called context in the case of theorem provers, that gives a good starting point to what we consider to be the context of a mathematical session. It will be clear that the context is highly dynamic: for instance, if, in the example of the matrix A above, the user wants to consider the matrix over $\mathrm{GF}(11)(x, y, z)$ rather than $\mathbb{Q}[x, y, z]$, a change of the coefficient ring should be the corresponding action on the context.

So, what is needed is a way to exploit such context data whenever a server providing a service to the user needs more knowledge. We have only made a modest beginning with the study of context, and we foresee that substantial research is needed for a successful implementation.

3 The Mathematical Document Server

In the previous section we have explained how we envision smooth communication using OpenMath with several mathematical services on the web. These services will enrich mathematical documents considerably. In this section we present a general model for a highly interactive mathematical environment embodying such features.

The heart of our architecture is a *mathematical document server*. In our approach, this server takes input from mathematical source documents, mathematical (web) services and users, and serves a view on the interactive doc-

ument to the user. The document server takes care of the presentation of the document to the user, it handles the communication between user and several mathematical services. It also manages the (mathematical) context in which presentation and communication take place. Figure 3 displays the essential parts of the proposed architecture and their dependencies.

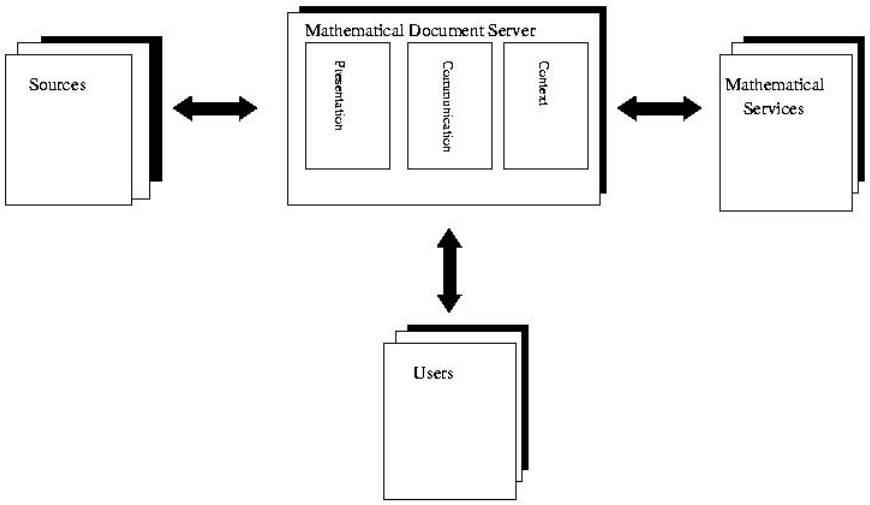

Fig. 3. General architecture

The architecture we have chosen for an interactive mathematical document is based on the idea of an interactive book: the (static) mathematics is included in a source document (or *source*, for short). It is highly structured and semantically sufficiently rich to create an exact mathematical description of the content and to allow actions (see also Subsection 4.2). The source document indicates the kind of action that is supposed to be offered in the interactive environment. The actions themselves however are realized by the document server. The document source is written by an author.

The document server on the other hand, is written by tool developers. It uses the mathematical content from the sources together with input from the user and mathematical services to specify the *context* of the interactive document. The context certainly depends on user actions; a jump from one entry in the source to another may alter the context. But also results from queries to mathematical services are input to the creation of the context. Within this context a presentation of the content relevant to the mathematical setting in which the user 'resides', is realized and can be presented to the user via an interface. In line with the discussion of §2.2, the context consists of

- assignments to variables of OpenMath objects (interpreted as definitions),
- OpenMath objects representing mathematical assertions,
- logistic information, for example, the user's id, mathematical background, permissions to use commercial services, etc.

At any given time, the context gives a precise description of the state the user is in by means of this data.

A simple example of this model is realized in a LaTeX environment. Here the document server produces, on user's demand, a dvi or postscript file from a LaTeX source and serves it using a dvi or postscript viewer to the user. Here the context is just given by the user's request to create a dvi or postscript file to view on the screen or send it to a printer, etc. In this simple example, the logistic data are relevant, but not the mathematical context.

A more advanced example can be realized in an XML-Java setting. Here the source consists of an XML-source. The document server creates, for example by XSL-transformations, an Html or XML document and serves it as a web page to the user. Using a web browser as an interface, the user can view a presentation of the document. Interactivity and communication with mathematical services can be realized inside the web server using Java-applets or servlets. Our present approach to realizing interactive mathematical documents is based on this example and will be discussed in the next section.

Creation and bookkeeping of the context as well as presentation of the content is taken care of by the document server. This server also handles *communication* between the source, the user and mathematical services. It stores presentation information and the context as dynamic data. The context is relevant in communication with the outside world. Using the model of communication with a mathematical service, the provider of the service may be aware of the context of the user's mathematics. This can take place by means of incremental steps (loading the context at the initial stage and translating the user defined changes one by one), or by means of downloading the entire context (or relevant portions thereof) upon receipt of each new query.

Of course, our primary target is a mathematical context, where, for example, in a chapter on ring theory of an algebra book, the field of coefficients might be specified to be a finite field. By interaction of the user interface with the user, this context can be further specialized to, say, the field of order eleven GF(11).

In an example on irreducible polynomials in a polynomial ring, the document server will take care of choosing the polynomial ring $GF(11)[X]$ over GF(11). Within this context the reader can now verify that the polynomial X^2+2X-2 is reducible, whereas the polynomial X^2+2X+2 is irreducible.

Some variables in the context can also be of a logistic nature. For instance the user name might help to create or recognize an individual version of the

context. This might be of relevance, for instance, for tracking the way a student reads an interactive text book.

4 MathBook, our implementation

The discussion of §3 regards our conceptual framework for interactive mathematical documents. In this section we discuss our progress in implementing this architecture within a JAVA-XML set-up. Our motivating example is a forthcoming new edition of the interactive book *Algebra Interactive!* (see [7]), which is interactive course material for undergraduate algebra. We shall use the word *MathBook* for the ensemble of software tools we are building for the construction of interactive mathematical documents such as, but not limited to, 'Algebra Interactive'. The distinct components of the architecture are displayed in Figure 4. We shall deal with them separately.

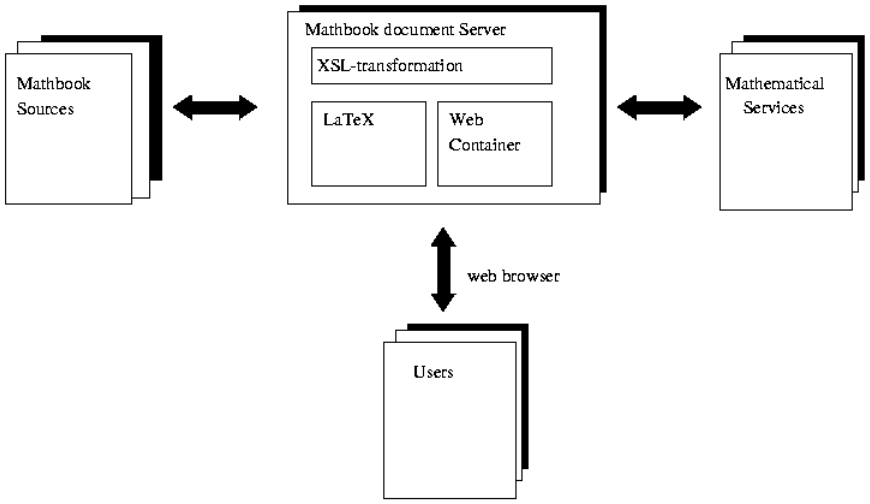

Fig. 4. MATHBOOK implementation

4.1 The MathBook Source

We have derived our own experimental grammar in the guise of a document type definitions (DTD) for the MATHBOOK source, an XML document. As a result, there is an XML based markup language (the MATHBOOK DTD) for the creation of interactive mathematical documents.

In creating a DTD for MathBook, we have been influenced by both DocBook [8] and OMDoc [24]. The former is a fairly general standard for structuring (into chapters, sections, etc.) electronic documents, the latter is a very rich, and strongly logic-oriented standard for mathematical documents. We intend to maintain a close link with OMDoc, but found the overall machinery involved too heavy for our purposes. The connection with DocBook is of importance to us, since we expect several authoring tools for it to emerge in the coming years, tools that could be of use to us in one form or another. MathBook deviates in various respects from OMDoc and DocBook and contains new features, like support for actions.

The mathematics in the source is given by means of OpenMath objects. This feature has clear advantages in terms of portability. The DocBook type grammar sees to it that there are natural scopes, where mathematical objects 'live'. For instance, when a chapter begins with "Let $\mathbb{F}$ be a field", the scope of the variable $\mathbb{F}$ is assumed to be the whole chapter (although, somewhere further down the hierarchy, say in a section of the chapter, this assignment can be overridden).

```
<OMOBJ>
 <OMA>
  <OMATTR>
   <OMATP>
    <OMS name="pres" cd="ida"/> <OMSTR "frac"/>
   </OMATP>
   <OMS name="divide" cd="arith1"/>
  </OMATTR>
  <OMA>
   <OMS name="plus" cd="arith1"/>
   <OMA> <OMS name="divide" cd="arith1"/> <OMI>3</OMI> <OMI>4</OMI> </OMA>
   <OMA> <OMS name="divide" cd="arith1"/> <OMI>2</OMI> <OMI>3</OMI> </OMA>
  </OMA>
 </OMA>
</OMOBJ>
```

Fig. 5. An attributed OpenMath object.

The mathematical content is represented in OpenMath. This means that the semantics is taken care of satisfactorily, but that no attention is being paid to presentation. In general, this is in line with the idea that presentation should be taken care of by the document server rather than the source. There are however some clear exceptions. Let us give two examples. In LaTeX, for each individual fraction, the author has a choice between a slash and a fraction display. In

$$\frac{3/4 + 2/3}{5}$$

we have used both. The other example concerns the statement "$3, 4 \in \mathbb{Z}$". The corresponding OpenMath expression would be the equivalent of "$3 \in \mathbb{Z}$ and $4 \in \mathbb{Z}$", whereas the presentation in the first form is highly desirable from an esthetic point of view.

In order to have such a flexible presentation, we are using presentation annotated OpenMath. This means, that in our MathBook source we allow style attributes inside OpenMath objects. Figure 5 shows an attributed expression corresponding to the LaTeX macro `\frac`, in which the fraction display is forced. It is assumed here that the default presentation is 'slash' so that the two other `divide`s need not be attributed. Of course, the author can also force the slash presentation of these by a similar attribution. By discarding these style attributes, regular OpenMath is obtained. So, one can easily go from annotated OpenMath to 'bare' OpenMath.

Within the MathBook grammar, special attention is also given to interactivity. For this purpose a whole range of tags (that is, structuring elements defined in DTDs and appearing in XML documents between pointed brackets) have been introduced. We will give two snippets of code appearing in the MathBook source to illustrate some of these tags and to give the reader an idea of how an author may create interactivity.

```
<eval scope="session">
   <OMOBJ>
    <OMA>
     <OMS cd="univpoly1" name="expand"/>
     <OMA>
      <OMS cd="univpoly1" name="gcd"/>
      <getomcontent> <getvarvalue name="poly_a"/> </getomcontent>
      <getomcontent> <getvarvalue name="poly_b"/> </getomcontent>
     </OMA>
    </OMA>
   </OMOBJ>
</eval>
```

Fig. 6. Some MathBook tags for interactivity.

The code in Figure 6 uses the combined effects of the following tags.

- getvarvalue: Read two strings representing OpenMath objects that were previously stored in the variables `poly_a` and `poly_b`, respectively. Note that the scope is set to session. This implies that, at the time the user visits this particular part of the source, the context will have the variables poly_a and poly_b, but when the user leaves it, these will no longer stay alive. The scope is introduced by a command like
  `<enablescope scope="session"/>`
  (not displayed in the figure). The above code then creates an OpenMath object (in fact, a univariate polynomial) that is placed in the session scope.
- getomcontent: Get the content of the OpenMath object, i.e., remove the markers `<OMOBJ>` and `</OMOBJ>` at the beginning and the end of an OpenMath object.

- eval: this tag indicates that in a realization of the source as an interactive document, the constructed OpenMath object is sent to a computational backengine, like *Mathematica*, for evaluation.

```
<addtoscope name="matrixsquared" scope="session">
<OMOBJ>
 <OMA>
  <OMS cd="arith1" name="times"/>
  <getomcontent> <getfromscope name="matrix"/> </getomcontent>
  <getomcontent> <getfromscope name="matrix"/> </getomcontent>
 </OMA>
</OMOBJ>
</ida:addtoscope>
```

Fig. 7. Some MathBook tags for content control.

The snippet in Figure 7 shows how objects in a context can be created. Here, the OpenMath object named `matrix` is read from the session scope (by means of the `getfromscope` tag). This object is used to create a new OpenMath object that is placed in the session scope (by means of the `addtoscope` tag) with name `matrixsquared`.

4.2 The MathBook server

As mentioned in Section 3, the mathematical document server, called the MathBook *server*, should cater for presentation, communication, and context in our implementation.

One part of our MathBook server consists of an XSL transformer together with a set of XSL stylesheets. The transformer picks up the MathBook sources and transforms them into LaTeX files for creating printouts or JavaServer Pages (JSP) to create an interactive realization of the source. JSP technology [17] is designed to develop dynamic web pages easily. A JavaServer Page is a template containing standard Html code, user defined tags, and Java scriptlets encoding the logic and the required behaviour of the dynamic web page.

The JavaServer Pages together with our Java tools form a Web application residing in a Web container of a standard (JSP/Servlet) server, allowing us to extend the functionalities of any such server. Note that, as opposed to Web*Mathematica*, we are not modifying the server itself.

So far, besides ROML (discussed at the end of §2.1), our Java tools include phrasebooks and a (JSP based) tag mechanism for bringing to life the interaction specified in the MathBook source. The phrasebooks deal with the following kinds of action.

- Sending an OpenMath object to a backengine (e.g. Mathematica) for evaluation and returning an OpenMath object.

- Retrieving the answer to a mathematical query from a web service and reacting on the outcome.
- Transforming OpenMath into MathML presentation.

```
<phreval id="result" name="GapPhrasebook" method="EVAL" scope="session">
 <OMOBJ>
  <OMA>
   <OMS cd="integer1" name="factorof"/>
   <OMI> <expression>b</expression> </OMI>
   <OMI> <expression>a</expression> </OMI>
  </OMA>
 </OMOBJ>
</phreval>
<if> <condition> <expression>result</expression> </condition>
 <then>
  <para> Answer:
   <expression>b</expression> divides <expression>a</expression>.
  </para>
 </then>
 <else>
  <para> Answer:
   <expression>b</expression> does not divide <expression>a</expression>.
  </para>
 </else>
</if>
```

Fig. 8. Retrieving an answer.

As an example of the first two features, consider the code displayed in the beginning of Figure 8. This is part of a JSP page obtained from a MathBook source. There, the OpenMath object contained in the `phreval` tags is sent to the GAP Phrasebook and passed to GAP for evaluation, and the response from GAP and the GAP Phrasebook is an OpenMath object (a Boolean) assigned to the variable `result` which lives inside the scope called `session`.

Communication is governed by Java servlets and phrasebooks; the actions defined within the MathBook sources are mapped onto and taken care of by a Java tag library, called the MathBook *tag library*.

The tag library is the tool that allows a practical implementation for actions in MathBook grammar. As a side issue, we mention that the library can also be used independently by someone who is just interested in developing his/her own JSP pages.

In the above example, the phrasebooks are invoked by the `phreval` tag. Furthermore, the tag mechanism handles

- the flow within a document, like if/then/else (see Figure 8), for loops, etc.
- the context. For example, objects can be stored and retrieved from the context.
- Casting OpenMath objects to OpenMath objects of another (often more structured) kind. This mechanism is especially useful for software systems

that do not type their objects very strongly. For instance, GAP does not distinguish a list of lists from a matrix, so when we expect a matrix to be returned (a fact that is noticeable in the presence of a context), we cast the list of lists onto a matrix of the right kind. Observe that this cast indeed belongs to the MathBook server and not to the source.

In particular, the tags in the library deal with two of the three features of a mathematical document server: communication, and context. The remaining feature, presentation, is dealt with by a separate Java program that translates the OpenMath objects of the source into MathML. It keeps track of the attributes for presentation such as the one for fraction discussed in §4.1. The MathML appearing in a JSP page can be rendered by a browser like Mozilla or Amaya, cf. [23,1].

Experimental MathBook servers for 'Algebra Interactive', with both *Mathematica* and GAP as backengine, have been realized. We intend to experiment further with CoCoA, Maple, and formal automated proof assistants such as COQ.

At the moment we are investigating the possible extension of the tag library in such a way that we are able to interface with other more geometry or visualisation oriented packages. We have started work on interfacing to the geometry package Cinderella [29].

4.3 Examples

We make the picture described above more concrete by considering three scenarios. In each of these scenarios a suitable interactive mathematical document can offer the appropriate mathematical services via the internet. In [12] we have described a way to provide the computing facilities of various mathematical software packages via OpenMath servers to the internet community. However, in these scenarios the mathematical services cannot consist solely of web interfaces with computational backengines, but ask for more specialized activities.

1. An author of a book on mathematical analysis has used both Maple [19] and *Mathematica* [20] to write algorithms discussed in his/her book. At times, the results of one system are fed into the other. Preferably, the author would like to write the code only once.

By use of the MathBook tools, the algorithms written in either system can be made to run virtually within the electronic version of the book. The MathBook tag library then takes care of communication with the backengines Maple and *Mathematica* and the computer algebra code is stored in the source. An alternative to sending native code to the backengines is to work with OpenMath. If the commands are confined to standard applications such as factorization of polynomials, the EVAL interpretation of the phrasebooks suffice (currently, a Maple phrasebook based on ROML does not exist,

but one based on [26] could probably be used). In this case, we can express input and output as well-understood mathematical objects (using the cast of the tag library, if necessary); moreover, we can ask for an evaluation by any third computer algebra system for which a phrasebook exists. For the author, the implementation has been reduced to writing a simple `phreval` tag. There is a third option in which more elaborate commands can be run on back engines. It uses a first version of an algorithm CD. We expect to be able to write most of the 130 gapplets (i.e., interactive examples using GAP) from the former edition of 'Algebra Interactive', in this OpenMath code, in such a way that each of the systems GAP, CoCoA and *Mathematica* will be able to run the code at the server end.

2. At a high school, students have been assigned a project on cryptography. In this context, information about prime numbers is required. They need to know the definition of a prime number, to find a few prime numbers of 200 digits and to compute related encoding and decoding keys.

There are many home pages about prime numbers, see e.g. [27] for an interesting one. Most of these sites contain a lot of static information, but lack available computation power, for instance to check primality of a given number. As part of the ESPRIT OpenMath project, we have made sample pages on prime numbers, backed up by GAP and *Mathematica*, in which the students can actually profit from the computation power of these backengines without having to know anything from the syntax of these backengines. They can retrieve primes with 200 or more digits to build a realistic and safe RSA cryptosystem, they can break such systems using too small primes, they can search for Mersenne primes, etc.

Upon request, calling a 'Pocklington' server (cf. [6]), the students can obtain a full proof (in words) of the primality of a given number. This works in much the same way as the stabilizer subgroup example discussed in §2.2.

3. Cinderella [29] is a beautiful program for exploring traditional planar geometry. Configurations can be constructed corresponding to classical theorems such as Pappus' Theorem, in which the conclusion is that three points are on a line. Once the configuration of points and lines is drawn, the fact will present itself 'automatically'. The user can drag and rescale the configuration while the three points stay on a line. An OpenMath representation of this configuration, using the OpenMath CD 'plangeo1', see Figure 2, can be translated into a formal description, a presentation in words, or to Cinderella input. Conversely, Cinderella (at least an experimental version) is able to provide such OpenMath expressions. We intend to explore these interactions further in collaboration with Ulrich Kortenkamp.

5 Conclusion

We have argued that OpenMath objects suffice to communicate mathematics in a rigorous way between software systems, but that two more features are

of immediate need: query facilities and management of the context of the mathematics in which the user is immersed. A solution of the query problem seems feasible on a fundamental level within the OpenMath framework, but the context problem requires more experimentation. We expect that the MathBook tag library will solve some of the most urgent matters in this respect. Authors can use it to augment their xml sources (in MathBook format), so as to obtain a high degree of structured mathematical interactivity.

A major obstacle to authoring an interactive document is the inaccessibility of xml source code, the enormous number of brackets and labels, such as in the examples of code in Figure 8. The general expectation is that, once good special purpose editors have been developed, no author will need to work with the elaborate xml sources. However, currently there is no alternative at hand. There are two editors for OpenMath objects, viz. [18,22], but these do not suffice for the more elaborate source documents described in §4.1. By means of the MathBook tag library, we have tried to reduce the difficulties of authoring as far as possible, but some xml editing remains necessary.

Another issue to be explored is 'searching for mathematical content'. Standard xml techniques might work on CDs. Although the interdependence of CDs is rather loosely organized (there is a `CDUses` field in a CD indicating on which other CDs the definition of symbols contained in it depend), standard xml tools will be able to produce the dependence trees. For example, via the `CDUses` construct we will be able to unravel that `times` in the core CD `group` refers to `times` in the core CD `monoid`, which in turn refers to `times` in `arith1`. Mathematical knowledge always has a hierarchical structure. It is the question whether the `CDUses` construct will suffice for an efficient implementation of the full hierarchy and the related searches.

References

1. Amaya, W3C's Editor/Browser, `www.w3.org/Amaya`.
2. Axiom interface to OpenMath. OpenMath ESPRIT Deliverable, 2000, `www.nag.co.uk/projects/OpenMath/final/node10.htm`.
3. A. E. Brouwer. *Coding theory server, for bounds on the minimum distance of q-ary linear codes,* $q = 2, 3, 4, 5, 7, 8, 9$, `http://www.win.tue.nl/~aeb/voorlincod.html`.
4. O. Caprotti, A. M. Cohen, and M. Riem. *Java Phrasebooks for Computer Algebra and Automated Deduction.* SIGSAM Bulletin, 2000. Special Issue on OpenMath.
5. O. Caprotti and A.M. Cohen. *Connecting proof checkers and computer algebra using OpenMath,* pp. 109–112 in *The 12th International Conference on Theorem Proving in Higher Order Logics* (Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, L. Théry eds.) Nice, France, September 1999. Springer Lecture Notes in Computer Science, vol. 1690.

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. A.M. Cohen, H. Cuypers, H. Sterk. *Algebra Interactive!*, Interactive lecture notes on Algebra (paper book and CD-Rom), Springer-Verlag, Heidelberg, August 1999.
8. DocBook, `http://www.docbook.org`.
9. H. Barendregt and A.M. Cohen. *Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants.* J. Symbolic Computation **32** (2001) 3–22.
10. O. Caprotti, A.M. Cohen, D. Carlisle, *The OpenMath Standard*, `www.nag.co.uk/projects/omstd/`.
11. O. Caprotti, A. M. Cohen, H. Cuypers, H. Sterk. *OpenMath Technology for Interactive Mathematical Documents*, to appear in Lisbon Proceedings.
12. O. Caprotti, A. M. Cohen, H. Cuypers, M. N. Riem, and H. Sterk. *Using* O*penMath Servers for Distributing Mathematical Computations,* pp. 325–336 in: ATCM 2000: Proceedings of the Fifth Asian Technology Conference in Mathematics, Chiang-Mai, Thailand, Wei Chi Yang, Sung-Chi Chu, Jen-Chung Chuan (eds.), ATCM, Inc., 2000.
13. A.M. Cohen, H. Cuypers, H. Sterk (eds.). *Some Tapas of Computer Algebra*, Springer-Verlag, Heidelberg, 1999.
14. A.M. Cohen and L. Meertens. *The ACELA project: Aims and Plans,* pp. 7–23 in *Computer-Human interaction in Symbolic Computation* (ed. N. Kajler), Texts and Monographs in Symbolic Computation, Springer-Verlag, Wien, 1998.
15. J.C. Faugère's Polynomial Equations Server, `www-calfor.lip6.fr/~jcf`.
16. GAP interface to OpenMath. OpenMath ESPRIT Deliverable, 2000, `www-groups.dcs.st-andrews.ac.uk/~gap/Info4/deposit.html`.
17. JavaServer Pages, for dynamically generated Web content, `java.sun.com/products/jsp/`.
18. Jome: Java OpenMath editor, `http://mainline.essi.fr`.
19. Maple, the computer algebra system, `www.maplesoft.com`.
20. *Mathematica*, the computer algebra system, `www.wolfram.com`.
21. MathML, Mathematical Markup Language, `www.w3.org/TR/MathML2/`.
22. MathWriter, Stilo's editor for rapid generation of mathematical expressions for display and processing on the web (handles MathML and OpenMath), `STILO: www.stilo.com`.
23. Mozilla, a browser development project, `www.mozilla.org`.
24. OMDoc, a standard for open mathematical documents, `www.mathweb.org/omdoc/`.
25. OpenMath Society Website, `www.openmath.org`.
26. PolyLab Java Phrasebook for Maple, `team.polylab.sfu.ca/~warp/openmath0.7.6.tar`.
27. Prime Pages, `http://www.utm.edu/research/primes`.
28. ROML, The RIACA OpenMath Library, `crystal.win.tue.nl/public/projects`.

29. J. Richter-Gebert and U. Kortenkamp. *Cinderella. The interactive geometry software* (book and CD-Rom), Springer-Verlag, Berlin, Heidelberg, 1999. See also `www.cinderella.de/en/index.html`.
30. N.J.A. Sloane. Online Encyclopedia of Integer Sequences, `www.research.att.com/~njas/sequences`.
31. A. Solomon, C.A. Struble. *JavaMath: an API for Internet accessible mathematical services,* to appear in Proceedings of the Asian Symposium on Computer Mathematics (2001), `www.illywhacker.net/papers/ascm.ps`.
32. A. Solomon, *Distributed Computing for Mathematical System Integration,* to appear in this volume, 2001.
33. Tomcat, servlet container used in the Jakarta Project, `jakarta.apache/org/tomcat`.
34. Unicode version 3.2, including virtually all of the standard characters used in mathematics, `www.unicode.org/unicode/reports/tr25`.
35. Webmathematica, Mathematica on the Web, `www.wolfram.com/products/webmathematica`.
36. R. A. Wilson. Atlas of Finite Group Representations, `www.mat.bham.ac.uk/atlas`.
37. `www.w3.org/XML/Query`.