Interactive Mathematical Documents on the Web

A.M. Cohen H. Cuypers E. Reinaldo Barreiro
H. Sterk
Department of Mathematics
Eindhoven University of Technology

January 21, 2002

Abstract

This paper deals with our work on interactive mathematical documents that make use of the World Wide Web. The work is concerned with a computer environment that is able to invoke various specialized mathematical software systems.

We are concerned with interactive mathemtical documents taking input from various sources, users and mathematical services. Communication between these three different entities has been realized using OpenMath. But, such communication and the interactivity inside the mathematical document take place in a specific, but dynamic context or state. In this paper we present a solution on how to keep track of this dynamical state and describe our implementation of this.

1 Introduction

Although the notion of an interactive mathematical document has been around for several years, cf. [13], its realization is nowhere near the final stage. Recent web technological progress, for instance, has enabled a much 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, and a mathematical service on the internet can be created by building an interactive mathematical document. The purpose of this paper is to describe an architecture and the latest tools we have developed towards an interactive mathematical document. In this vein, the paper can be viewed as a sequel to [11].

In this paper we describe the type of web service we have in mind, as well as a way in which we have realized such services. Thereby we focus

on the user end of this window to the world of mathematical services and show how interactive mathematical documents may be served to users. In particular, we describe a *mathematical document server* which takes input from various sources, mathematical services and users to create a context dependent and highly interactive document and serves it to the user.

We would like to stress that the technology for achieving such goals already exists. We give a description of these in the setting of Java Technology: Java Beans, Applets, Servlet and JSP technology,XML Java Technology and our own Java Technology. 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.

2 A Framework for Interactive Mathematics

In this section we describe the notions involved in our approach to interactive mathematics. In §2.1, we begin by overviewing OpenMath. Next, in §2.2, we address two additional requirements for our purposes: query facilities and a notion of state of the mathematics that is being communicated. Then, in §2.3, we sketch the architecture of an interactive mathematical enviroment.

2.1 OpenMath

Starting point for semantically rich communication across the Web is the standard for mathematical expressions OpenMath, cf. [23]. This representation of mathematics 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, a sort of XML dictionaries of mathematics. The standard documents and the collection of public Content Dictionaries for OpenMath are available from [23]. For a further introduction to OpenMath, see [11]. There it is also explained how OpenMath and MathML complement each other in that OpenMath objects express mathematical content whereas MathML mainly focus on presentation. The connection between the two rests upon the facts

- that MathML works with a relatively small number of commonplace mathematical constructs chosen within the K-12 realm of applications and

- that it provides the content symbol (`csymbol`) for introducing 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.

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 Content Dictionaries 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 Content Dictionaries 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. However, in communication, one often wants to convey more than just the mathematical objects by themselves. Although a message could be a mathematical object by itself (e.g., read "The generalized Riemann Hypothesis holds"), usually, one party asks a question and the other gives a response. Naturally, we would like a standard way of expressing queries, a management system for parameters accompanying the question, and a reference mechanism to couple a message to a query of which it is an answer.

Currently, certain OpenMath objects, especially constructs using 'application' are implicitly understood to be queries by means of interpretations given to them in phrasebooks. The latter are programs for handling OpenMath objects, parsing these into an application-native language (e.g., Mathematica, Maple, GAP), sending the result to the application, catching the response from the application and translating those 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 experimental versions of both Axiom and GAP, cf. [2, 15]. 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 [26]. 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

The OpenMath set up is only the beginning. For mathematical services across the internet, many more issues have to be resolved. In §3.2 of [28], some of these issues are listed. 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, but, although it will remain useful for clients to be aware of feasibility, they will be more concerned with the validity of the answer. Also, 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 we shall only address two matters of interest in this direction that arise from our bottom approach starting with OpenMath:

- how to express queries and
- how to take into account the state in which a mathematical query takes place.

We have argued that, so far, the OpenMath setup seems rather primitive in that only mathematical expressions are passed, and no indications as to what the required action on the object is. Currently, the phrasebook makes this interpretation, and so the matter is resolved by a declaration from the side of the phrasebook what its action (interpretation) is. 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 printer.

But there are probably better solutions, which come from automated proof checking. A slight extension of the language in which we formulate mathematical assertions will enable us to formulate mathematical queries. As noted in [9], type λ calculus expressions like $\Gamma \vdash ? : A$, where Γ represents the context (see below) and A an assertion of which a proof is requested, are expected to embed into a full type checking mechanism without problems (that is, the type inference is expected to remain a valid algorithm, and so on). So, within the OpenMath framework, we expect that it is possible and sound to handle queries by means of a CD defining the most fundamental types of question asked.

In the remainder of this paper, we mainly focus on the second issue, how to handle the state in which a mathematical query takes place. We note that this problem has not been addressed in some of the more successful mathematical services on the internet, such as Sloane [27], Faugère's Gröbner basis service [14], Wilson's Atlas of representations of finite simple groups [32], Brouwer's coding theory data base [3], and WebMathematica [31].

For example, Webmathematica is a way to access 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 CAS (=Computer Algebra System) point of view, the *state* 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[?] and Out[?] variables in Mathematica).

However, we wish to incorporate one more feature in our notion of state. This one is taken from logic, where the notion of a context is our source of inspiration. Indeed the symbol Γ above stands for context, and contains, besides definitions, statements, which are interpreted as 'the truth'. 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 contra-

diction. In particular, it would not be desirable to have the starting state 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 of statements, which is called context in the case of theorem provers, that gives a good starting point to what we consider to be the state of a mathematical session. It will be clear that this 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 state (or context).

So, what is needed is a way to exploit such data, regarding the state or context, whenever a server providing a service to the user needs more knowledge. In the remainder of this paper, we mainly describe how we intend to solve this issue.

Above we have explained how we envision smooth communication using OpenMath with several mathematical services on the web. In this way, we can enrich mathematical documents with mathematical services. However, an interactive mathematical document should allow several types of interaction, depending on its context and the input obtained from its source, a user or some mathematical services. This implies that the source of such an interactive document, should not only be semantically rich but also highly structured and enabling the interactivity.

2.3 The Mathematical Document Server

In this section we specify a general model of how to create highly interactive mathematical documents.

The heart of our architecture is a *mathematical document server*. In our view, this server takes input from mathematical source documents, mathematical (web) services and users and serves a view on the interactive document 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 and keeps track of and manages the (mathematical) context in which presentation and communication take place. Figure 2 displays the essential parts of our architecture and their dependencies.

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 descrip-

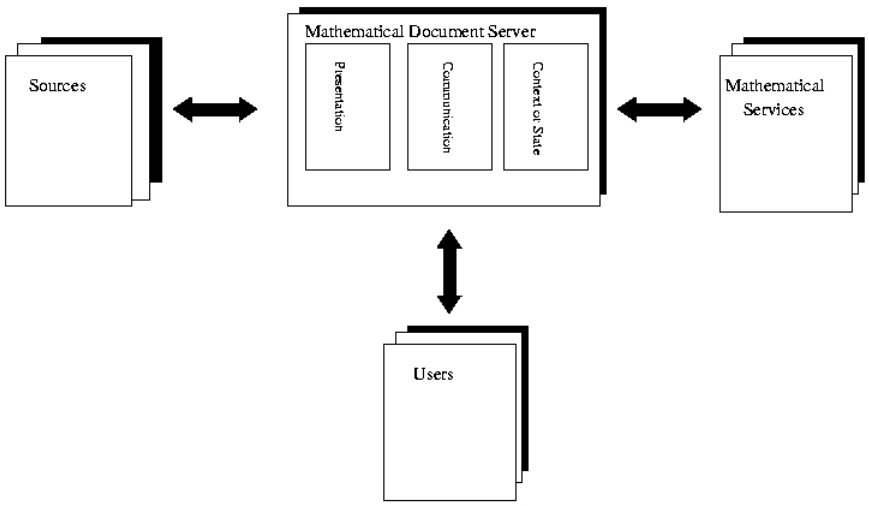

Figure 1: General architecture

tion of the content and to allow actions. The document server uses this mathematical content together with input from the user and mathematical services to specify the *context* or *state* 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 form input for 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 (and so are interpreted as definitions), or
- OpenMath objects representing mathematical assertions.
- results from queries obtained from mathematical services.
- but also logistic information, for example, the users id, his mathematical background, his permissions to use commercial services, etc.

At each 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 enviroment. Here the document server produces, on demand from a user, 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 a postscript file, to view on the screen or send it to a printer, etc.

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 server to the user. Using a web browser as 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 servlet. Our present approach for 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 user document server. This server also handles the *communication*, between the source, the user and mathematical services. It has presentation information and the context stored as dynamic data. The context is relevant for the mathematics in the communication with the outside world. Depending on the model of communication with a mathematical service, the provider of the service may be aware of the state the user's mathematics is in 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 the relevant portions thereof) upon receipt of a 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 GF(11) of order eleven. In the presentation of an example of a polynomial ring, the user interface will take care of choosing a polynomial ring over GF(11).

Some variables in the context can also be of a logistic nature. For instance the user name might help to create a personalized version of the context, relevant for tracking the way a student goes through an interactive text book.

3 Mathbook, our implementation

In this section we discuss how we have implemented the architecture presented in §2 within a JAVA-XML enviroment. Our big motivating example

is a forthcoming new edition of the interactive book *Algebra Interactive!* (see [7]), which is interactive course material for first year undergraduate algebra. We shall use the word *MathBook* for an interactive mathematical document, as well as for the ensemble of software tools we are building for the construction of interactive mathematical documents such as Algebra Interactive. We shall deal with each of the components of the architecture separately.

3.1 The Mathbook Source

We have derived our own 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. We have been influenced by both DocBook [8] and OMDoc [22]. The former is a fairly general standard for electronic books, 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. Also, the connection with DocBook is of importance to us, since we expect several authoring tools for it to emerge in the coming few years, and we want to profit from their presence.

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). Within the MathBook grammar, special attention is also given to interactivity.

3.2 The Mathbook document server

As mentioned in the previous section, the mathematical document server, in our approach called the *Mathbook document server*, should cater for presentation, communication, and context. It should support a wide range of *actions*: ways to easily define interactions with other mathematical (and non-mathematical) services. Actions are important because they enable the true interactivity for the user. Examples of actions are:

- sending an OpenMath object to evaluate to a backengine (e.g. Mathematica),

- retrieving the answer to a query from a web service and reacting on the outcome.
- transforming OpenMath into MATHML presentation,
- casting OpenMath objects to OpenMath objects of another (often more structured) kind (for instance, a list of lists onto a matrix),
- searching within documents for mathematical content,
- placing (and retrieving) objects into the context,
- controlling the flow within a document,
- iterating actions,
- making a printout of the document

In our current implementation, we have realized this by using a JAVA web container/server (e.g., tomcat or BEA weblogic) including or in conjunction with an XSL-transformer.

The XSL-transformer uses Mathbook XSL style sheets to transform the Mathbook sources into web applications (XML and JSP pages, together with JAVA-software to be discussed below) that reside on the web container. This web server presents these documents to a user via a JAVA-enable browser. The web browser also acts as a user interface to the web server. The communication between web container, user and mathematical services is controlled by JAVA technology contained in the web container. Indeed, 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 taglib*. We elaborate on these in the following subsections.

As we also want to be able to print our documents, we have also developed XSL style sheets transforming our sources into LaTeX. By adding a LaTeX enviroment to our Mathbook document server, we can produce the desired high quality printouts.

3.3 Presentation

In a way, the first order of business is to make the content of the source visible to the user. There are numerous ways to prepare for such *views*. The generic way in which we prepare a view is by XSL-transformation of the XML Mathbook source using dedicated XSL style sheets. We have developed

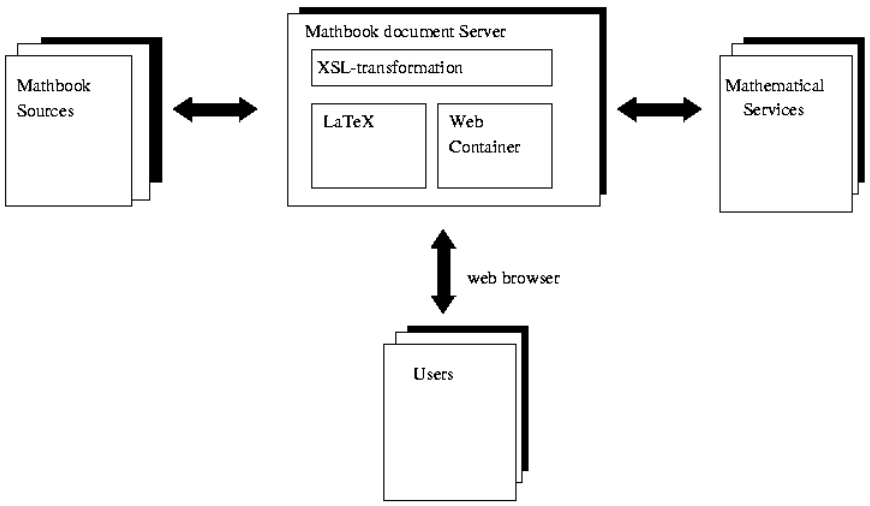

Figure 2: Mathbook implementation

XSL style sheet to transform our sources in LaTeX and HTML or presentable XML. In the latter case OpenMath objects are translated into MATHML, the standard for displaying mathematics over the internet.

In interactive sessions, we have a dynamical situation, so that it is desirable to transform OpenMath objects on the fly. Because the programs working with XSL stylesheets are interpreters, this is a slow process, therefore we have also written a OpenMath to MathML phrasebook. This JAVA software performs the translations directly and resides inside the web container.

The OpenMath to MATHML phrasebook behaves very much like any phrasebook, albeit that sometimes we need a little help from the source beyond regular OpenMath. Let us give two examples. In LaTeX, for each individual fraction, the author has a choice between a slash and a fraction display. In

$$\frac{3/4}{5/6} = \frac{3/2}{5/3} = \frac{9}{10} = 9/10$$

we have used them 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 flexible presentation, we are using presentation annotated OpenMath. This means, that in our Mathbook source we allow style attribute inside OpenMath objects. By just discarding these style attributes, regular OpenMath is obtained. So, one can easily go from annotated OpenMath to "bare" OpenMath.

3.4 Communication

All our communication tools are based on Java Technology. The ROML based phrasebooks (cf. §2.1) operate as web components in the web applications inside the JAVA enviroment of the web container and enable us to make mathematical software packages accessible over the Web. There are examples of such services in our experimental version of Algebra Interactive for both Mathematica and GAP. We intend to experiment further with CoCoA, Maple and theorem prover engines such as COQ.

3.5 Context

The existence of a context is meaningful only if the author can install powerful tools for the user to keep track of inputs and for using this information to check consistency and to update pages automatically when these are logically related. With these observations in mind, we developed a first version of context management within the Mathbook tag library at the web container.

We will give two snippets of code to illustrate some of the tags of the Mathbook taglib and to give the reader an idea how an author may create interactivity. The code

```
<ida:xslt url="http://localhost:8080/mathml/xsl/omobj-mathml.xsl">
 <ida:cast type="univpoly">
  <ida:phreval name="MathematicaPhrasebook" scope="session">
   <OMOBJ>
    <OMA>
     <OMS cd="univpoly1" name="expand"/>
     <OMA>
      <OMS cd="univpoly1" name="gcd"/>
      <ida:getomcontent>
       <ida:getvarvalue name="polynomial_a"/>
      </ida:getomcontent>
```

```
      <ida:getomcontent>
       <ida:getvarvalue name="polynomial_b"/>
      </ida:getomcontent>
     </OMA>
    </OMA>
   </OMOBJ>
  </ida:phreval>
 </ida:cast>
</ida:xslt>
```

uses the combined effects of the following tags.

- getvarvalue: Read two strings representing OpenMath objects that were previously stored in the variables polynomial_a and polynomial_b, respectively.
- getomcontent: Remove the markers `<OMOBJ>` and `</OMOBJ>` at the beginning and the end of an OpenMath object.
- phreval: Send the constructed OpenMath object to Mathematica for evaluation. More precisely, the `phreval` tag reads its content (a string object) and makes use of a Mathematica JavaBean named MathematicaPhrasebook, that was previously created and stored in the session scope, to translate it to Mathematica syntax and send it to Mathematica. Once an answer is obtained from Mathematica, the corresponding OpenMath object is generated and translated to a string object containing the XML encoding of the OpenMath object.
- cast: Cast the returned OpenMath object, representing an algebraic expression, to an OpenMath object representing a univariate polynomial.
- xslt: Produce the MathML presentation encoding for the OpenMath object (by means of a style sheet). More in detail, the `xlst` reads the OpenMath object, that the `phreval` tag returned, and transforms it into MATHML by means of the XSL style sheet specified on the *url* attribute of the `xslt` tag.

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 polynomial_a and polynomial_b, but when the user leaves it, these will no longer stay alive. The scope is introduced by a command like

`<ida:enablescope scope="session"/>`. The above code then creates an OpenMath object (in fact, a univariate polynomial) that is placed in the session scope.

The next snippet shows how objects in the context can be changed.

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

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*.

The ability to store session bound objects allows us to implement context tracing tools. Servlet technology has the ability to maintain server side objects in different scopes. By default, it supports scopes for the application, the session, the page, and the request. For interaction with backengines, as we have seen, the most important one is the session scope. Since for each user a different session scope is created, the server side can distinguish between different clients (users). The state of the user's mathematics can be loaded by reading in all the variables defined in the session scope.

Let us come back to the example of the chapter that begins with "Let $\mathbb{F}$ be a field". It is very easy for the author to create a page with an example computation within an effective field (that is, a field in which the arithmetic is feasible on a computer) of the user's choice. The specific choice of $\mathbb{F}$ will be eradicated when the user closes the example.

3.6 Examples

We make the picture described above more concrete by considering the following five scenarios.

1. An author of a book on mathematical analysis has used both Maple [18] and Mathematica [19] 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.

2. At a high school, students have been assigned a project on cryptography. In this context, they need to know about prime numbers. 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.

3. An engineer of an international oil company visits a platform to repair one of the crucial pumps. At the spot, the engineer needs to know the outcome of some involved computations regarding the flow in pipes connected to the pump, the input values of which are read off from the pump's gauges.

4. A financial consultant, visiting a client, connects a laptop to a server of some investment banks or insurance companies and uses the most up to date information of the stock market and of the client as input for computations leading to a consultancy.

In each of the above 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 the scenarios described above, the mathematical services cannot consist solely of web interfaces to some computational backengines, but ask for more specialized activities. The engineer of an oil company in Scenario 3 does not really need to know the outcome of some involved computations regarding the flow in pipes connected to the pump, but needs to relate these results to the specifications of the pump. He might want to know which parts of the pump should be replaced.

In Scenario 4, the financial consultant is not really interested in the outcome of the mathematical queries, but might want to present to the client those life insurance options best fitting the circumstances. The high school student studying prime numbers in Scenario 2 not only wants to get large prime numbers from a backengine, but wants to use these numbers in some of the examples of an RSA cryptosystem, without knowing the syntax of the backengine.

To finish the discussion of our implementation, we revisit the five scenarios and indicate what can be achieved with the MathBook tools.

1. *An author of an interactive book on mathematical analysis has used both Maple [18] and Mathematica [19] to write algorithms discussed in his/her*

book. 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 IDA 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 [24] 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 CAS 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, they need to know about prime numbers.* There are many home pages about prime numbers, see e.g. [25] 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 form the computation power of these backengines with having to know anything from the syntax of these backengines. They can retrieve primes with 200 or more digits to built a realistic and safe RSA cryptosystem, they can break such systems using too small primes, they can search for Mersenne primes, etc.

3. *An engineer of an international oil company visits a platform to repair one of the crucial pumps.* In the MathBook software it is straightforward to write an interactive document for the purpose that the Mathbook document server fetches the data needed by the engineer, link it to mathematical software for the relevant computations, and retrieve the required integers or floating point numbers. By means of the JAVA technology, the document presented to the engineer may only display the conclusion (e.g., which gauge should be replaced) in words and/or pictures.

4. *A financial consultant, visiting a client, connects through his/her laptop to the main servers of some investment banks and companies and uses the most up to date information of the stock market and of the client as input for computations leading to a consultancy.* The Mathbook document server may now be on the laptop of the cosultant. It takes care of retrieving the important information from various financial sources and combines this with the wishes of the client. Then the data is translated into the right syntax to be used by the mathematical software for calculating an optimal solution for the clients problems. The financial consultant need not be concerned with the composition of two programs. Within a session scope, the data fetched from the main server of the companies can be held, and prepared for input into the private computations.

4 Conclusion

We have argued that OpenMath objects suffice to pass on mathematics in a rigorous way between software systems, but that two more features are of immediate need: query facilities and management of the context (or state) 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 IDA 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 amount of brackets and labels, such as in the examples of code in §3.5. 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. [17, 20], but these do not suffice for the more elaborate source documents described in §3.1. By means of the IDA 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 has always 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/\Om{}/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 Arjeh 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] Olga Caprotti, Arjeh M. Cohen, Hans Cuypers, Hans Sterk. *OpenMath Technology for Interactive Mathematical Documents*, to appear in Lisbon Proceedings.

[12] Olga Caprotti, Arjeh M. Cohen, Hans Cuypers, Manfred N. Riem, and Hans Sterk. *Using OpenMath 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 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

[14] J.C. Faugère's Polynomial Equations Server, `www-calfor.lip6.fr/~jcf`.

[15] GAP interface to OpenMath. OpenMath ESPRIT Deliverable, 2000, `www-groups.dcs.st-andrews.ac.uk/~gap/Info4/deposit.html`.

[16] JavaServer Pages, for dynamically generated Web content, `java.sun.com/products/jsp/`.

[17] Jome: Java OpenMath editor, `http://mainline.essi.fr`.

[18] Maple, the computer algebra system, `www.maplesoft.com`.

[19] Mathematica, the computer algebra system, `www.wolfram.com`.

[20] 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`.

[21] Mozilla, a browser development project, `www.mozilla.org`.

[22] OMDoc, a standard for open mathematical documents, `www.mathweb.org/omdoc/`.

[23] OpenMath Society Website, `www.\om{}.org`.

[24] PolyLab Java Phrasebook for Maple,
team.polylab.sfu.ca/~warp/\om{}0.7.6.tar.

[25] Prime Pages, http://www.utm.edu/research/primes.

[26] ROML, The RIACA OpenMath Library,
crystal.win.tue.nl/download/.

[27] N.J.A. Sloane. Online Encyclopedia of Integer Sequences,
www.research.att.com/~njas/sequences.

[28] Andrew Solomon, *Distributed Computing for Mathematical System Integration,* to appear in this volume, 2001.

[29] Tomcat, servlet container used in the Jakarta Project,
jakarta.apache/org/tomcat.

[30] Unicode version 3.2, including virtually all of the standrad characters used in mathematics, www.unicode.org/unicode/reports/tr25.

[31] WebMathematica, Mathematica on the Web,
www.wolfram.com/products/webmathematica.

[32] R. A. Wilson. Atlas of Finite Group Representations,
www.mat.bham.ac.uk/atlas.