Using OpenMath Servers for Distributing Mathematical Computations

O. Caprotti[*] A. M. Cohen H. Cuypers M. N. Riem H. Sterk

Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
`{olga,amc,hansc,mriem,sterk}@win.tue.nl`

Abstract

This paper presents *OpenMath* servers and their role in an architecture for distributing mathematical computations. Such an architecture is needed in the context of mathematical problems whose solution is best achieved by use of techniques from different areas. We describe such a setting integrating various computational engines. Finally we provide case studies that illustrate this technology.

1 Introduction

In recent years, the possibility of distributing mathematical computation is becoming a reality using internet technology. Mathematical problem solving often requires the use of techniques from different areas, but individual mathematical software packages do not always cover the methods needed: general purpose packages cover a wide range of basic algorithms whereas most specialized packages only deal with algorithms applicable in a restricted field. When using the computer, both in teaching and research, there is a need for implementing a framework where all computational resources are made available and are fully integrated.

This paper concentrates on the architecture we have implemented to distribute mathematical computations based on *OpenMath* servers. The *OpenMath* standard language provides a semantically rich representation of mathematics for communicating mathematics between software packages [12], in particular it is general enough to allow the integration of computational and deductive machinery [6]. We describe the architecture and give examples in the context

[*]This author was supported by the OpenMath Esprit project 24969

of teaching material, e.g. *Algebra Interactive!* [8], and of research oriented problems. There are a number of related approaches, several of which use *OpenMath* [2, 19] in a client-server setting. Some of these are in the design stage, others restrict the scope of the implementation to computer algebra software. Our approach has been tested for distributing problems to various software packages. In fact, it is designed to meet the requirements of a future general agreement on the protocol to be used for mathematical servers (e.g. IAMC, KQML, OpenXM). For example, problems on the borderline of group theory and linear algebra can be tackled using, say, the group theory package GAP and the general purpose package Maple. In interactive teaching material, like a basic algebra course covering number theory, polynomials and groups, various topics depend for computational aspects on the use of different back engines.

The implemented architecture is based on *OpenMath* servers on a network able to provide mathematical services on *OpenMath* objects. A client in charge of solving a problem accesses these servers with specific computational requests understood by the different packages to which the servers interface. Before the request is sent to the server, the *OpenMath* objects involved in it are translated to the appropriate syntax and package-syntax is used for the dispatched command. The answer received is translated back into an *OpenMath* object which is returned to the client.

The paper is structured as follows. Section 2 very briefly recalls the basics of the *OpenMath* language. *OpenMath* Servers are described in Section 3 where in particular *Phrasebooks* play an important role. Section 4 reports on the available JAVA software that can be used to implement *OpenMath* servers. Three case studies on distributing mathematical computations among *OpenMath* servers are shown in Section 5. A concluding section contains the final discussion.

2 *OpenMath*

The *OpenMath* standard language [12] provides a semantically rich representation of mathematical information for electronic access and usage. Here we limit the exposition on *OpenMath* to some examples of mathematical objects that occur later in the paper. The reader is referred to the standard documents available from [20] for the details.

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

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

OpenMath Content Dictionaries (CD) collect and provide definitions of mathematical notions for usage within *OpenMath* applications. The official repository for CDs is [20]. The *OpenMath* symbols in the object above that identify the polynomial ring structure obtained from the integers modulo p are `polyr:PolynomialRingR` and `setname2:Zm`. More precisely, they are the symbols called `PolynomialRingR` and `Zm` defined in the CDs `polyr` and `setname2`, respectively.

A polynomial in this ring, say $f = X^3 - X + 1$, can be represented in several ways as an abstract *OpenMath* object, for instance by using the `polyr:PolynomialR` constructor for recursive polynomials. As with all *OpenMath* objects, it can be encoded in a human-readable format using XML [13] and stored as:

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

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

3 *OpenMath* Servers

In this section we define what we mean by *OpenMath* server as contrasted to a generic mathematical server interfaced to a single back-engine. Essentially the difference lies in the usage of *OpenMath* to facilitate mathematical computations possibly using a combination of several back-engines.

3.1 Mathematical Servers

In this paper, mathematical servers refer to servers handling requests invoking computational aspects of a mathematical nature, e.g. symbolic integration or proof verification. There is yet no standard accepted protocol for mathematical servers; some of the commercial packages have started to provide custom ways of turning their software into a mathematical server. The approach taken here is independent of the package acting as back-engine. This is common to other efforts such as the internet accessible mathematical protocol by P. Wang [19]. Agent technology using the Knowledge Query Manipulation Language KQML is also used in designing intelligent networks of mathematical servers in which the user need not be aware of which mathematical server performs the request.

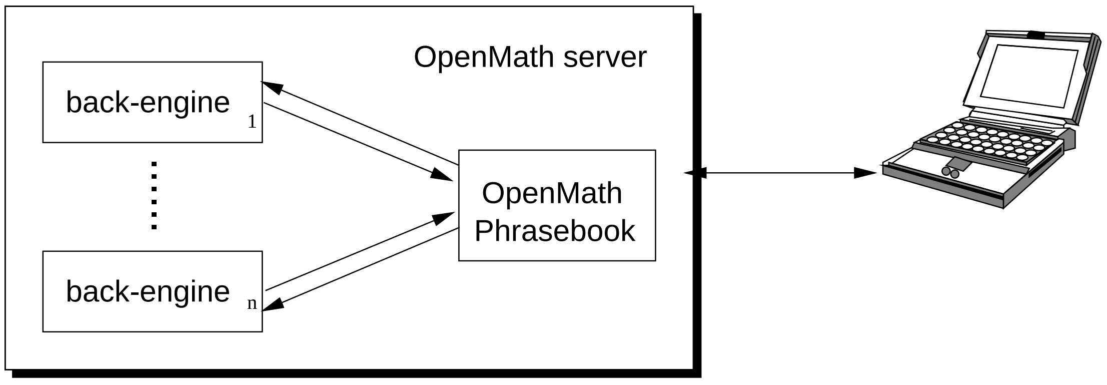


Figure 1: *OpenMath* Mathematical Server

3.2 *OpenMath Phrasebooks*

In this section we concentrate on *OpenMath* servers of which one vital ingredient is the *OpenMath Phrasebook*. *Phrasebooks* are programs that interface the back-engines to the client via *OpenMath*, as shown in Figure 1. The definition of *Phrasebook* in the *OpenMath* standard is informal. Here, because of the applications we have in mind, we need a more precise description of a *Phrasebook*.

The *Phrasebooks* we designed are defined in terms of a 4-tuple: a list of CDs, control information, interpretation, and communication.

OpenMath-compliancy requires that a *Phrasebook* declares the list of CDs it recognizes.

Control information, e.g. EVAL, SIMPLIFY, PROVE, SOLVE, PRINT, expresses the tasks that the *Phrasebook* is able to perform. This aspect falls outside the definition of *Phrasebooks* in the *OpenMath* standard where the default task is assumed to be evaluation (EVAL) as is customary in computer algebra software. For our purposes, since we want to use also back-engines which are capable of carrying out a variety of tasks, the control information needs to be explicitly available.

Interpretation refers to what is going to be done to the objects using the back-engines. Interpretation is a function of the control information and the received *OpenMath* objects. Some recent formalized approaches on how to specify mathematical services provided by *OpenMath Phrasebooks* are found in [2].

Finally, the *Phrasebook* specifies how the actual *communication* between the software package and the *OpenMath* computer environment is achieved. In our experiments we have used TCP/IP communication. In general, the *Phrasebook* implementor is free to choose any communication protocol.

Although a *Phrasebook* can only function properly if it is specified in some detail, there is as yet no guarantee that the actual implementation of the *Phrasebook* conforms to its specification.

As a first example, consider a client submitting the request to solve the equation $x^2 = 2$

over $\mathbb{R}$ to an *OpenMath* server powered by a *Phrasebook* specified by the 4-tuple consisting of:

CDs	`arith1`, `relation1`, ...
Control Information	SOLVE
Interpretation	$[\text{SOLVE}, \textbf{application}(\texttt{relation1:eq}, A)] = B$ where B is a real solution of the given equation
Communication	`Maple:mathweb.org:4217`, `Mathematica:localhost:4126`, `PoSSo:posso.dm.unipi.it:4564`

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

If all goes well, an *OpenMath* object is produced by the codec. For the equation at hand, the *OpenMath* object is

$$\textbf{application}(\texttt{relation1:eq}, \textbf{application}(\texttt{arith1:power}, x, 2), 2) \qquad (2)$$

The *Phrasebook* decides, following the specification given by the interpretation, which action to take on the *OpenMath* object. In case of ambiguous requests, that is several equivalent actions could be taken, user-intervention can be required to choose the desired action. So assume that the action is known and thus also the back-engines which are to be used for the computation. The *Phrasebook* prepares the queries for the selected back-engines by using the interpretation for distributing the computations. In simple cases, the control information corresponds directly to a single query (certain user-commands) in a single back-engine. In general however, the *Phrasebook* is able to extract from the control information and the *OpenMath* object an algorithm that splits up the problem and distributes the solving process among several back-engines. See for instance the architecture of the primality proof generator *Phrasebook* that uses both proof checker Coq and GAP [7]. The queries sent to the back-engines consists of expressions in the back-engine syntax. These are produced in part by calling the codec's decoder on the relevant *OpenMath* fragments. Communication happens at the shell interface of the various back-engines because this is the easier solution for turning third-party software into a server in absence of ad-hoc solutions, e.g. most software packages do not provide a callable library of functions. In some special cases, proprietary tools can be used for setting up the communication, e.g. JLink/MathLink for Mathematica. The *Phrasebook* assembles the results it has received and produces *OpenMath* output from them. This output is displayed/sold to the user in a suitable form.

3.3 Architecture

A possible architecture to realize the process described in the previous paragraph needs a few more ingredients that take care of input and output. We base our examples on the usage of browsers for accessing the computational resources. In that case, JAVA applets and/or servlets are a convenient means of implementing the graphical user interface. Figure 2 depicts the general scheme.

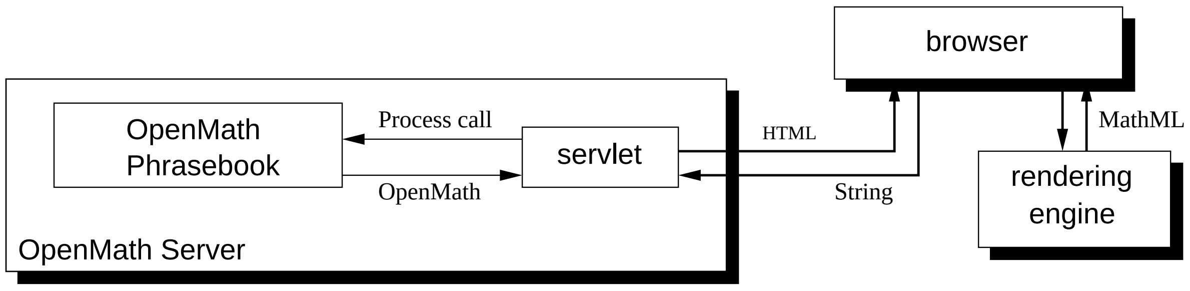


Figure 2: Exemplary Architecture

A web page describes the problem and includes input fields. The user may enter the data in a format selected among several options (e.g. GAP or Maple syntax). If the input fields are given in an HTML form handled via a servlet, then the input data is processed and translated by a JAVA program into the suitable process call understood by the *OpenMath Phrasebook*. The process call must specify the control information expected by the *Phrasebook* and the input data in terms of *OpenMath* objects. A menu-based graphical interface can be used to enumerate choices or data examples.

The servlet also receives the output from the *Phrasebook* and is in charge of producing a document to present the result. In general this document is an HTML document containing calls to the rendering engine for displaying the mathematical fragments it contains. The next generation of browsers is expected to be able to natively support MathML-presentation, thus making a separate rendering engine superfluous.

4 *OpenMath* Java Software

Internet technology is often realized in JAVA. Because the architecture outlined in the previous section is internet-based, we have developed *Phrasebooks* and *OpenMath* servers using JAVA. This approach is not uncommon, to wit an interface based on *OpenMath* to various packages for computer algebra has been put forward within the CATHODE project [3] and a web interface to Maple's visualization capabilities has been worked out by [14].

Official *OpenMath* libraries are distributed by the *OpenMath* Society. Best known are the Naomi (*OpenMath*) JAVA library by the PolyMath Development Group [17] and the JAVA and C libraries by INRIA [11]. The libraries provide classes to manipulate *OpenMath* objects in the XML and the binary encoding. Additionally, the libraries include encoders and decoders to/from arithmetical expressions written in Mathematica or Maple syntax. Similar translation features exist for various other languages, for instance for MathML [4], and for LaTeX [9].

For the software developments described in the next section, we have written JAVA classes for the *Phrasebooks* implementing the architecture described above [15]. The client-server communication is TCP/IP based. These *Phrasebooks* are easily extended to suit specific demands. Furthermore, we have added some new codecs to the ones distributed with the Naomi library. They translate the syntax of the proof checkers Lego and COQ, and of the computer algebra packages Mathematica and GAP.

5 Case Studies

Algebra courses provide typical examples where at present no single computer algebra package supports all computational aspects without programming efforts on the side of the user (instructor). Topics ranging from arithmetic to (polynomial) rings to (symmetry) groups are best supported by distinct packages. Although packages tend to extend their scope, thus possibly resolving the need for distinct packages in a given specific setting, developments within mathematics may establish links between seemingly unrelated fields, whence creating situations for which probably no single package is readily available. For example, the new field of Algebraic Statistics [16], in which Gröbner basis methods play a role in the design of experiments, links two subjects hitherto far apart in the mathematical spectrum. Hence, Algebraic Statistics is a natural place for combining back-engines from the fields of algebra and statistics. In the following subsections we describe in more detail a few instances where distributing mathematical computations can be used.

5.1 Teaching with *Algebra Interactive!*

Algebra Interactive! [8] is interactive course material for first and second year university algebra. It covers basic arithmetic, modular arithmetic, polynomials, arithmetic modulo a polynomial, permutations, monoids and groups, rings and fields, and permutation groups. It comes on a CD-rom with an accompanying book; it is viewed through a browser and contains extra navigation tools in addition to the standard navigation options. The material is enlivened by JAVA applets and so-called gapplets. JAVA applets provide many dynamic illustrations with a strong visual aspect as well as calculators pertaining to the relevant material. Gapplets are applets that interface to the computer algebra package GAP [10] and provide illustrations of computational aspects in input/output form. GAP specifically aims at group theory computations and is therefore not the natural choice for other topics covered. Currently, a second edition of

Algebra Interactive! [8] is under construction that interfaces to more (freeware) back-engines, viz. CoCoA [5] and GAP at present, that belong more naturally to the individual chapters.

The following is an example where the chapters on arithmetic modulo polynomials and on permutations overlap, but where the corresponding back-engines differ: CoCoA for polynomial computations and GAP for the group theory computations. Consider an irreducible polynomial $f \in \mathbb{Z}_p[X]$ of degree n, where $\mathbb{Z}_p$ denotes the field with p elements, p prime. Then the ring $\mathbb{F} = \mathbb{Z}_p[X]/(f)$ is a finite field with p^n elements. If $a \in \mathbb{F}$ is an invertible element, then multiplication by a determines a permutation $\sigma_a : \mathbb{F} \to \mathbb{F}$ of the finite set $\mathbb{F}$. To study the group theory aspects of such permutations, it is natural to transport them to GAP, being a back-engine better suited for dealing with permutations. For instance, questions like: what is the order of σ_a?, or: what type of group is generated by permutations of the form σ_b, with b running through a set of invertible elements?, can be easily solved by GAP. Conversely, to find an invertible element realizing a given permutation (and deciding if this is possible at all) requires a computation better suited for CoCoA. In Section 2 we have shown examples of how the relevant objects are represented in *OpenMath*.

Experimental *OpenMath* CDs, called `permut1`, `group1`, and `permgrp`, for representing permutations and group theoretical notions are available from the *OpenMath* website [20]. The *OpenMath Phrasebook* supporting the interaction needed to solve this kind of problem handles questions regarding polynomials by calling CoCoA or Maple (for instance the generation of the p^n elements in $\mathbb{F}$) and questions regarding permutations by GAP.

5.2 Algebraic Statistics with CoCoA and R

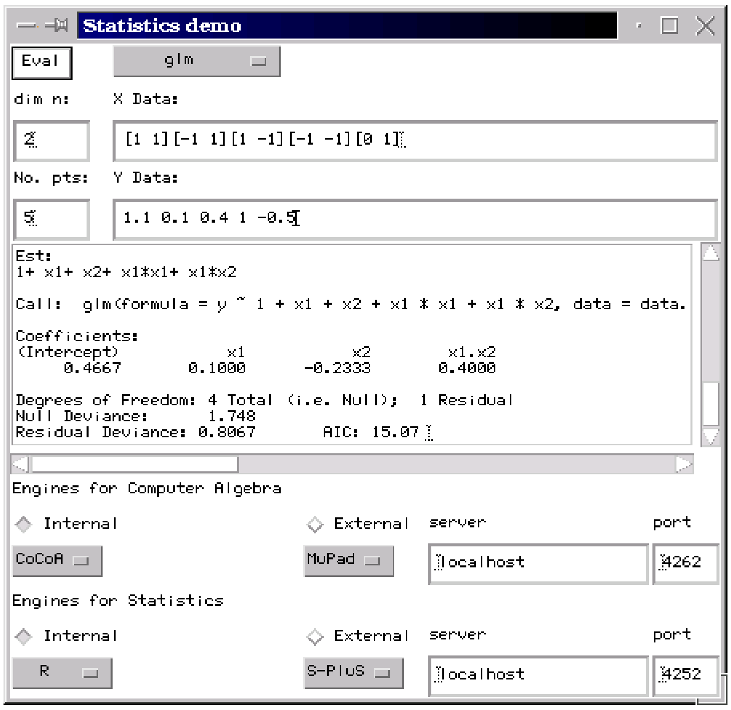
In the following we outline a set-up where both algebraic and statistical methods are ingredients in dealing with design of experiment. The basic problem is to identify a model explaining certain data associated to a finite (fractional factorial) design $\mathcal{F}$, given as a subset of some $\mathbb{Q}^n$. Any function on $\mathcal{F}$ can be described as the function associated to a (not unique) polynomial. Finding a model is then redirected into a problem concerning polynomials. Gröbner bases can be used to determine a set $\mathcal{S}$ of monomials (the 'support') out of which a polynomial supporting a model is to be constructed. Given such a set of monomials, there may be reasons from statistics to actually construct the model using a subset of $\mathcal{S}$. For example, higher order interactions, like terms of the form X^mY^n with exponents $m, n \geq 2$, are sometimes omitted since they may lack statistical meaning. Finally, to identify the 'best possible' model for the given data set and with the support just constructed, a regression method from statistics is invoked.

For the sake of brevity, we omit other subtle considerations in this case study and focus on the heart of the matter.

This set-up leads to the following. The first item refers to computations in the realm of algebra, and can be done naturally in software packages handling polynomial arithmetic. For our experiments we have chosen CoCoA [5] since it has some of the relevant algorithms readily available. The second item is of a statistical nature and requires the use of a statistical package. We have used the freely available software R [1].

Start with a finite design $\mathcal{F} = \{P_1, P_2, \dots, P_m\} \subset \mathbb{Q}^n$, and with a data set. These are stored as *OpenMath* object representing a matrix of integers (the CD `linalg2` provides the symbols for doing it) and an array of *OpenMath* floating point numbers.

- Choose a term order σ and compute the associated standard set of power products $\mathcal{P}_\sigma(\mathcal{F})$ using, e.g. the Buchberger-Möller algorithm (see [16]). In CoCoA this amounts to first using the function `IdealOfPoints` on the appropriate translation of the finite design and then selecting the proper monomials. Since selection of the monomials has to be programmed ad hoc, it could be carried out as well in a different computer algebra system.
- Suitable regression methods construct a model out of the $\mathcal{P}_\sigma(\mathcal{F})$ and the data set. Any statistical package dealing with regression is suitable for such an analysis.



dim n: 2
X Data: [1 1][-1 1][1 -1][-1 -1][0 1]
No. pts: 5
Y Data: 1.1 0.1 0.4 1 -0.5

```
Est:
1+ x1+ x2+ x1*x1+ x1*x2

Call:  glm(formula = y ~ 1 + x1 + x2 + x1 * x1 + x1 * x2, data = data.

Coefficients:
(Intercept)          x1           x2         x1.x2
     0.4667      0.1000      -0.2333        0.4000

Degrees of Freedom: 4 Total (i.e. Null);  1 Residual
Null Deviance:      1.748
Residual Deviance: 0.8067       AIC: 15.07
```

Computer Algebra: server localhost, port 4262
Statistics: server localhost, port 4252

Figure 3: Statistics and Computer Algebra

Figure 3 is a screenshot of our experiments in connecting *OpenMath* servers for solving problems in statistical design of experiments with the aid of algebraic methods. The user is prompted to input the dimension n, the number of points m, the set of points and the data set. By selecting from a menu, it is then possible to request an algebraic computation, for instance one that returns the monomials to be used in the construction of the fitting model.

Having obtained this estimate (in the screenshot this is what appears after "Est:" in the output window), the user can try to use it for fitting the data set. The menu allows to select general linear regression fitting by the option `glm`. This choice triggers the setup of a statistical computation request built from the input data and from the algebraic result obtained in a previous algebraic computation. The statistical package then receives a call in the proper syntax. Notice in particular that the codecs used in the *OpenMath Phrasebook* take care of producing the different syntax needed for CoCoA and for R when using the data set of points or the monomials.

5.3 Pure Mathematical Research with LiE and GAP

In the software package LiE [18], elements of a simple complex Lie group are given (up to conjugacy) by means of numerical data. For instance, $[1, 0, 2]$ would describe an element of order two in a group of Lie rank two. By means of an interface to the software package GAP, we can turn this data in to a 'real life' element of the Lie group, for instance in its realization as a group of automorphisms of the corresponding Lie algebra. For, in GAP, the complex simple Lie algebras have been implemented by Willem de Graaf and their automorphisms are linear transformations of the underlying vector space. So, after feeding the vector of integers describing an element of finite order from LiE to GAP, an easy function in GAP will produce the automorphism as a linear transformation, and its matrix or further relevant data regarding the automorphism could be fed back to LiE. This is just one step in the process of describing the finite (Lie primitive) subgroups of the Lie groups of exceptional type; the research of describing all such subgroups has been completed (up to conjugacy questions), but the subgroups are not yet readily available on computer.

6 Concluding Remarks

This paper describes the details of the technology and the architecture we are currently developing to distribute mathematical computations. This work is heavily based on *OpenMath* servers built around the *OpenMath* standard language and the *OpenMath Phrasebooks*. Because *OpenMath* provides a semantically rich representation of mathematics, it can be safely used for communicating mathematics between software packages. In particular, it is general enough to allow the integration of software from various areas of computational mathematics as we have demonstrated by the case studies shown in this paper.

Acknowledgments. The authors would like to thank Eva Riccomagno for some clarifications on Algebraic Statistics.

References

[1] *An Introduction to R*, `http://www.r-project.org/`.

[2] *Proceedings of calculemus-2000, 8th symposium on the integration of symbolic computation and mechanized reasoning*, 2000.

[3] M. Berth, F.-M. Moser, and A. Triulzi, *Implementing Computational Services Based on OpenMath*, Submitted, `http://paul.math-inf.uni-greifswald.de/Cathode2/omws/`.

[4] S. Buswell, *STARS*, Available from Stilo Technologies, April 1999, `http://www.stilo.com`.

[5] A. Capani, G. Niesi, and L. Robbiano, *Cocoa, a system for doing computations in commutative algebra*, Available via anonymous ftp from `cocoa.dima.unige.it`, 4.0 ed., 2000.

[6] O. Caprotti and A. M. Cohen, *Connecting proof checkers and computer algebra using OpenMath*, The 12th International Conference on Theorem Proving in Higher Order Logics (Nice, France), September 1999.

[7] O. Caprotti and M. Oostdijk, *How to formally and efficiently prove* prime*(2999)*, Proceedings of Calculemus 2000, St. Andrews, 2000.

[8] A. M. Cohen, H. Cuypers, and H. Sterk, *Algebra Interactive!*, Springer Verlag, 1999.

[9] OpenMath Consortium, *TEX to OpenMath Converter*, June 2000, Available at `http://www.nag.co.uk/projects/OpenMath.html` .

[10] ______, GAP *interface to OpenMath*, OpenMath ESPRIT Deliverable, 2000.

[11] ______, *OpenMath* JAVA *and C Library*, June 2000, Available at `http://www.nag.co.uk/projects/OpenMath.html` .

[12] The OpenMath Consortium, *The OpenMath Standard*, OpenMath Deliverable 1.3.3a, OpenMath Esprit Consortium, `http://www.nag.co.uk/projects/OpenMath.html`, August 1999, O. Caprotti, D. P. Carlisle and A. M. Cohen Eds.

[13] 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`.

[14] H. Le and C. Howlett, *Client-Server Communication Standards for Mathematical Computation*, Proceedings of ISSAC'99 (Vancouver, Canada), ACM, New York,, 1999, pp. 299–306.

[15] RIACA, *OpenMath* JAVA *Phrasebooks*, Eindhoven University of Technology, Available from `http://crystal.win.tue.nl/public/projects/`.

[16] L. Robbiano, *Gröbner Bases and Statistics*, Gröbner Bases and Applications (Bruno Buchberger and Franz Winkler, eds.), London Mathematical Society Lecture Note Series, vol. 251, Cambridge University Press, 1998, pp. 179–204.

[17] PolyMath OpenMath Development Team, *Java OpenMath Library: Version 0.7c*, `http://pdg.cecm.sfu.ca/openmath/`, June 1999.

[18] M. A. A. van Leeuwen, A. M. Cohen, and B. Lisser, LiE*: A package for lie group computations*, CAN, Amsterdam, 1992, URL: `http://www.can.nl/SystemsOverview/Special/GroupTheory/LiE/`.

[19] P. Wang, *Design and protocol for internet accessible mathematical computation*, Proceedings of the 1999 International Symposium on Algebraic and Symbolic Computation, 1999.

[20] OpenMath Society Website, *http://www.openmath.org*.