Style Sheet Languages and Mathematical Material

Richard M. Timoney*

May 26, 1998

Abstract

In this paper, we first survey some emerging technologies which are related to mathematics and computing, especially to mathematical communication via computer networks.

Before mathematics can be used 'interactively' on the World Wide Web, there are a number of obstacles that would need to be removed and we try to identify these. In particular we concentrate on style sheets for mathematics.

Contents

*This work was supported in part by the European Community under ESPRIT contract number 24.969.

1 Existing Technology

Under this heading LaTeX/TeX [11, 12] and Adobe PostScript together with its modern variant PDF (for Portable Document Format) [1] are the most significant from the point of view of mathematical documents. From the point of view of mathematics and computing, symbolic computation systems are reaching the centre stage as far as mathematical work and education are concerned but they are not yet at the same level of significance for the general public. On a more general front the Internet and its associated protocols are enabling many new developments, while more powerful processors are opening new possibilities very rapidly.

1.1 Mathematical Software

For the scientific and engineering world, Mathematica [28] and Maple [16] are perhaps the leaders in the symbolic systems field, with AXIOM [3], MuPAD [20] and Reduce [24] in a more minority category. At the more numerical end, MatLab [18] is probably the leading system and it is especially significant among engineers, though the free systems SciLab [25] and Octave [7] are also worth a look for those who are happy with less polished but free systems. None of the general purpose computer algebra systems are free now, though MuPAD is still available free for individual use. From the point of view of mathematical research there are a number of systems which aim to deal with restricted or specialised areas of mathematics GAP [8] (for group theory and more generally for discrete mathematics), CoCoA [4] (for computations in commutative algebra), Pari [23] (for number theory), Lie [13] (for computations with semisimple or reductive Lie algebras), Macaulay2 [10] (for commutative algebra and algebraic geometry), Magma [15] (for computational algebra, algebraic geometry and combinatorics). There are also several systems aimed at logic or theorem-proving which vary from being largely of mathematical interest to systems aimed at areas of theoretical computer science. There are several implementations of functional programming languages such as ML [19] (which is mainly of interest to computer scientists) and there are several theorem-proving systems (HOL [9], Coq [6], LEGO [14] for example) implemented using ML or variants of it. Nuprl [5] is another theorem-proving or proof-checking system which is based mostly on Lisp code. Many of these special-purpose systems are freely available.

From the point of view of a research mathematician, it could be a great advantage to have a common interface to all these systems and this essentially demands that they should all speak a common language. If they could speak a common language, one could harness several such systems in parallel to solve some problems more efficiently than one could do in a general purpose system such as Maple

or Mathematica. Alternatively, one could have more confidence in a result that depended in some way on computations with a computer algebra system if the computations had been checked with more than one independent system. Again, this would be facilitated if there could be a way to feed such systems information in a common language. The OpenMath [21, 22] effort aims to solve this problem and we will discuss that in more detail below.

From the educational point of view, one could imagine future textbooks with active mathematical formulae. At the moment, one can write Mathematica notebooks which achieve this provided the user of the material has the right version of Mathematica available or one could similarly write the textbook as a Maple worksheet. The ideal solution would work equally well with either system and would be still able to work with future versions of the software, or with other similar systems.

1.2 Electronic Publishing

From the publishing point of view, the current state of commercial electronic publishing is essentially to make material available electronically that would previously have been produced in paper form. The added value of the electronic medium is mainly provided by search facilities and ease of access. Companies are attracted by the advertising value of having their material available electronically and they are anxious to keep up with modern technology. From the point of view of individuals the HyperText Markup Language (HTML) used on the Web has provided exciting facilities for making new material available on the Web, and the excitement has lead many people to the view that the Web can revolutionise the world of scientific and scholarly publishing by reducing production and distribution costs and by enabling more interactive forms of material to be published. Publication of material in electronic form on CD-ROM has much the same attractions.

However, there are genuine obstacles to be overcome and it is not so clear that the end result will be any more dramatic than the invention of photocopiers has been over the past three decades.

Publishers and librarians can generally see worries about quality of material and about how to access todays electronic material with tomorrow's technology. As a salutary lesson we can point to microfiche as a breakthrough in storage technology which is now quite obsolete. Libraries with stocks of material in this form are faced with trying to transfer it onto newer media (or discarding it). In the computer world, constant evolution of technology has meant that standards for tape storage have been changing every few years.

As regards quality, the presentation of a high class journal or book has often been of a quality consistent with the content and publishers are conscious of the

value of a high standard of layout. HTML has advantages, but a high standard of layout is not one of them. Moreover, the fast evolution of the HTML language has meant that the same presentation cannot be guaranteed with last years version of a browser or even with the same browser on different hardware. Publishers are naturally unhappy with this and have tended to prefer Adobe PostScript as a medium for encoding their publications, though not as the original source encoding.

Search indices and other reference information must be supplied in addition to the PostScript code for individual journal articles or books. PostScript interpreters are freely available for many platforms (or are almost built in with Adobe Display PostScript on a few platforms) and high quality PostScript printers are widely available.

A more recent invention is the PDF language of Adobe and this is now being widely used for electronic publishing. It shares the advantages of PostScript but also has the ability to have internal hyperlinks (or internal active cross-references) and external hyperlinks (specified by a URL[1] as used on the Web) to other PDF documents or more generally to arbitrary documents on the Web.

A free reader know as `acroread` is distributed by Adobe as a stand-alone application and as a browser plug-in. Hence it can cooperate very well with a standard Web browser. Other freely available software (`ghostscript`, `gv` and `xpdf`) can also display and process PDF files and so there is no serious obstacle to the use of PDF for publishing — except a minor one that the user needs to download and install some PDF software.

PDF has advantages compared to PostScript, apart from the active hypertext links, and few essential limitations compared to PostScript. The text content is (for the most part) searchable in a PDF document and Web indexing software exists which is capable of indexing both PDF and HTML content. PDF allows files to be made so that they can be partially displayed while only part of the file has been read and various kinds of compressions to be used to cut down on the size of various portions of a PDF file.

From the point of view of mathematicians, LaTeX/TeX is still widely used as the input format for mathematical documents. It is widely regarded as producing the most satisfactory output. An impetus for the use of PDF for mathematical documents on the Web is provided by two factors:

1. A variant of LaTeX/TeX known as pdftex (and pdflatex) is now available as part of the standard distribution of the TeX system for UNIX systems. It allows LaTeX/TeX input files to be processed directly into PDF output documents with only cosmetic changes to the input. There are some limitations on pdftex, but a great advantage is that internal LaTeX cross references can

[1] Uniform Resource Locator

be transformed automatically into active PDF cross references merely by invoking the `hyperref` package.

2. No hypertext format which is generally accessible at the moment allows satisfactory rendering of mathematical formulae, apart from PDF.

However, there remain several imperfections with regard to PDF including:

- PDF is not a serious option as a source and archival format.
- Although PDF is interactive in so far as it allows hyperlinks and even has a form-filling interface, it is not suitable as a medium for interacting with mathematical documents.

Publishers have been moving towards the use of SGML[2] as the source and archive standard for their publication material. SGML enables documents to be labelled according to their structure (for example, the author(s), title, date, abstract, journal title, journal volume, sections, citations, and many more items can be all tagged by their purpose within a document). Thus SGML allows documents to be stored in such a way that the original document can be considered as both a database record describing the key features of the document and as a source form for publication in multiple formats (a journal article, a table of contents entry for a journal, an entry in an abstracting journal, data for a context-sensitive search engine, for example).

2 New Technologies

The SGML community have managed to have the SGML philosophy adopted by the World Wide Web Consortium in the form of a technical recommendation for XML (for eXtensible Markup Language) [29], which is essentially a simplified version of SGML. SGML demands that every document have a document type as specified by a DTD (Document Type Definition) and that each document should conform to the rules of its DTD. XML is similar except that there is greater freedom about how the DTD should be found and there is even a possibility of inferring the DTD from the structure of a given syntactically correct XML document.

It is intended that complete XML documents and HTML documents with embedded XML fragments will be viewable by future Web browsers in the way that HTML documents are now. To enable this several software developments will be necessary:

- Browsers will have to be able to parse XML documents.

[2] Standard Generalized Markup Language, an ISO standard 8879:1986.

- While the range of symbols accessible in HTML has always been severely limited (and discriminates against many languages) XML implies support for UNICODE, a system of character encoding that is close to universal in its scope. Browsers will need to display these characters.

- Browsers will have to be able to display and print XML documents in an acceptable way.

Style sheets are the proposed mechanism to address the last two points.

Given the emphasis put on XML in current discussions, the stated support for XML by the main companies involved in the Web and the number of new proposals based on XML that are appearing, one can expect that these problems will be solved in the relatively near future. A satisfactory resolution of these problems could well be important in enabling mathematics to be published on the Web in a way that has a greater interactive potential than PDF.

As of April 7, 1998 a technical recommendation for Mathematical material in XML format has been adopted by the World Wide Web Consortium. The recommendation is known as MathML (Mathematical Markup Language) [17]. In principle, the trend to XML should enable MathML to become a reality in the sense that it should become possible for the average user of the Web to view Mathematical material in an acceptable way.

However, a more detailed analysis of the problems points to the conclusion that the general XML and style sheet solution may not solve the problem sufficiently generally to cope with the requirements of MathML.

3 MathML and Style Sheets

3.1 Parsing MathML

MathML expects rules to be obeyed in addition to the rules which can be expressed in XML. This may make it more difficult to write tools to generate and validate XML, but as browsers support XML in general, they should support MathML.

MathML is a way of describing Mathematical formulae, and not whole documents. Thus an agreed way of embedding MathML in a wider context (such as HTML) is required, but again one can reasonably hope that a solution will be found in the general context of embedding XML fragments in (the next version of) HTML.

3.2 Font Problems

First, it is a challenge to support a wide range of fonts for Mathematics. Only the Adobe Symbol font (which is adequate only for rather basic mathematical usage) is available as standard and, up to now, browsers have not even provided clean support for that. It is possible to configure Netscape to display the Adobe Symbol font, but the characters must be coded in HTML by pretending that the Symbol font is font for ordinary text in the default encoding.

In principle a clean solution to these problems has been defined in XML, because it is stated that any UNICODE symbols can be used in XML and UNICODE is essentially comprehensive for most languages and also for mathematical symbols. In addition to having a clean mechanism to refer to strange symbols, it is necessary to overcome the problem of providing access to fonts in a form that can be used by browsers. If users have to install extra fonts on their computers to display pages containing mathematical symbols or text in eastern European languages, then it will continue to be a reason for not using the technology for material designed to reach a wide audience.

It is possible, however, that this obstacle can be removed by the technique of using fonts available on the Web. If combined with local caching of fonts used previously, and if the technology is good enough to allow satisfactory printing, then these problems should disappear in time.

3.3 Formatting Problems and Style Sheets

Formatting of mathematical formulae in an acceptable way requires a more complicated method of dealing with layout than is necessary for dealing with textual material, tabular material and external images.

Some existing solutions for displaying mathematical formulae on the Web treat each mathematical formula as a picture. This is not good for several reasons. The multitude of formulae in a typical mathematical document makes for a huge number of pictures for each page, resulting in slow loading times. Typically the pictures are provided in screen resolution and show up as granular when the whole page is printed. The pictures have a fixed size and do not adapt to global preferences in the browser such as changing the default font size to a larger one. An equally fundamental problem is that there is no agreed method for the graphic image for an embedded formula to communicate its baseline to the browser. The baseline is needed to align the formula properly with the text.

One could imagine future browsers calling Java code or other special code to format embedded MathML material, but unless the formatting of the Mathematical formulae is integrated with the formatting of the text most of the problems with using embedded images to represent mathematical formulae will remain.

A key ingredient in the movement to XML has been the idea that style sheets associated with XML documents will be able to define display or formatting rules for the documents. HTML had a loosely defined rule for how each type of element (heading level 1, heading level 2, paragraph, list, *etc.*) should be displayed, but XML allows the freedom to define new element names to suit the purpose in hand. Abstractly, that is good because it allows (for example) a price list to be published on the Web with the order numbers, descriptions, prices and VAT rate of each item flagged for what they are. One could decide on tag names `<ordnum>`, `<descrip>`, `<price>` and `<vatpct>` for these bits of data. The document on the Web could be identical to the database used internally in the company for the accounts.

The missing link is a rule to lay out the information in a way that the average person will be able to read conveniently. The solution to this is to have a style-sheet mechanism where formatting rules can be attached to element tags.

Such mechanisms exist already for HTML documents on the Web and allow one to vary the way a browser will deal with elements (for example to use an italic font instead of a bold font for headings). The current language for defining such style rules is called CSS (Cascading Style Sheets) and it is supported (at least in part) by the main browsers. However, it is relatively limited in its power. In particular it allows only a limited amount of context-sensitivity of the rules and does not allow elements to be displayed in an order different from the order in which they appear in the source document. One might want lists to be displayed differently if they are inside another list and differently again if they are twice nested. Such rules can be expressed in CSS by considering all possibilities of how one type of list can be inside another and giving rules for each one. A more natural way to specify such a rule would be to have a way to calculate the level of nesting and to be able to have a rule with a case statement or an if-else type rule depending on the result of the calculation.

There is a proposal for a new and more powerful style sheet language to be used on the web, known as XSL (for eXtensible Style Language) [30]. This can be expected to solve some of the problems, but the draft specification of XSL seems to be quite preliminary and it is not possible to tell how far XSL will go to solve the problems. As it stands, XSL has a scripting facility that is certainly powerful enough to deal with the example of nested lists as explained above. Similar problems arise with display rules for mathematics where it is often desirable that subscripts, numerators and denominators of inline formulae should be smaller than the usual font size but this size reduction should cease before the symbols become unreadably small if subscripts or superscripts are nested several times. Conventionally inline formulae are treated differently from displayed versions such as

$\lim_{x \to a} f(x)$ and

$$\lim_{x \to a} f(x)$$

for example. The simplest way to detect which rule should be chosen is to be able to find out if the limit is contained inside a display formula or not.

From this perspective, XSL seems to be satisfactory, but there is a major question as to whether XSL will support the so-called mathematical flow objects.

Proper formatting of mathematics requires that the browser should understand several constructs such as displaying fractions, building parentheses of various types around parenthesised expressions in such a way that the parentheses are just big enough to enclose the expression. A high degree of nesting is needed. This concept is known as the support of 'mathematics flow objects' in the formatting rules of the browser.

There is no real technical difficulty here. The problem is more of a political, practical and commercial nature. A theory of how this should work derives from an ISO standard language DSSSL[3] which is designed as the style sheet language for SGML. DSSSL is a very general language and actually has a core or essential part plus optional extras known as features. To make things more complex DSSSL is two languages, a transformation language and a style language (each with their own features or optional parts). The transformation language standardises a way of specifying rewrite rules for SGML documents of a given document type (for example to transform a document from one DTD to another where the tags have different names or have to be used in a different order). The style language is also a way of specifying transformation rules from a source SGML document to a formatting language. The result of processing an SGML document together with a DSSSL style specification for it is the same document written in a formatting language (such as RTF) and a further formatting is still required to produce appropriate line breaks, hyphenations and page breaks.

The problem with DSSSL is that there are few, if any, full implementations of DSSSL. A close approximation to an implementation of the DSSSL style language is a program called jade (freely available from URL `http://www.jclark.com`). This implements DSSSL together with the mathematical flow objects feature, but the number of output syntaxes is limited (RTF with WORDEQ for mathematical formulae is one possible output syntax).

DSSSL, together with the mathematical flow objects feature seems to be adequate to support MathML.

Hence, if it the necessity to support MathML is perceived as sufficiently pressing outside the inner circle of mathematicians, engineers and scientists, then the necessary parts of DSSSL could be included in XSL before it is finally adopted.

[3] Document Style Semantics Specification Language, ISO/IEC 10179:1996.

3.4 MathML and the DSSSL Query Language

However, a deeper analysis of the design of XML shows that, in addition to supporting the mathematics flow objects, XSL would also need to support another major part of DSSSL (which is not currently in the XSL draft) before it could format MathML in full.

MathML is a double solution of the problem of encoding mathematical formulae. It has a 'presentation' part which claims only to be sufficient to encode the information necessary for the layout of mathematical formulae and a 'content' part that aims to be able to also convey the mathematical meaning of formulae encountered in non-advanced mathematics. This doubling means that there are two ways to encode a fraction such as $\frac{a+1}{b}$. In the presentation version, this might be

```
<mfrac>
  <mrow>
    <mi>a</mi>
    <mo>+</mo>
    <mn>1</mn>
  </mrow>
  <mi>b</mi>
</mfrac>
```

A formatting rule for fractions as encoded this way should be able to be expressed in XSL, though it would need to concern every type of MathML element which could form the denominator or numerator of a `<mfrac>` element. For `<mrow>` the rule would have to include a condition about what to do if the `<mrow>` had a parent element of type `<mfrac>` and was moreover the first child or sibling. A different rule would be needed if `<mrow>` was the second child of an `<mfrac>` element. It should be possible (though not easy) to deal with all possible eventualities of this kind.

The alternative (content) coding of the same fraction could be:

```
<apply>
  <div/>
  <apply>
    <plus/>
    <ci>a</ci>
    <cn>1</cn>
  </apply>
  <ci>b</ci>
</apply>
```

The rules to display this correctly have to be much more complex, because it is not sufficient to consider parent elements or even ancestor elements. The `<ci>b</ci>` must figure out that it belongs to an `<apply>` or application element where the operation being applied to it is division (as determined by the type of the first element in the `<apply>` sequence). Similarly the `<ci>a</ci>` needs to figure out that it is part of a summation (but not last in the summation) so that a + can be drawn after it. This type of condition (based on the type of the first sibling of the element as well as the type of the parent) is not expressible in XSL. One could imagine using the scripting language associated with XSL to solve this problem by setting a stack of global variables as one passes the first child of an `<apply>` node and then popping the stack when one reaches an `</apply>` end tag. However, the draft XSL specification forbids such solutions on the grounds that they would be hard to implement. Indeed a little thought shows that such solutions would not work well in an interactive environment where a change of the type of the leading element in the `<apply>` would affect the formatting of all the other children of the `<apply>` element. Actually, the proposed solution is based on a batch mode model rather than an interactive model and if such scripts were allowed in general the formatting of the whole document might need to be changed because of one local change to the document.

An alternative method is to allow formatting rules to refer to the position of an element in a more general way. This solution is allowed in the DSSSL standard, but seems to add considerably to the complexity of DSSSL and it may be quite optimistic to think that XSL will support such a facility.

The author believes that this aspect of MathML did not receive adequate attention and that the design is unnecessarily difficult to implement in this respect. The 'presentation' part of MathML may well be implementable in XSL if the mathematical flow objects are included.

Of course, that does not mean that MathML cannot be implemented fully, but only that special extra code will be needed to support MathML and the general XSL style-sheet mechanism for XML cannot be expected to suffice. If there are many other XML DTDs that are defined in a similar way for other special purposes, and if all need some special-purpose code then one cannot expect that general browsers will support them all. MathML may well lose out along with other similar ventures.

3.5 Editing MathML

A problem that will also need to be solved before MathML becomes widely used is the problem of inputting formulae in MathML. Though it is theoretically possible to write the raw MathML tags by hand (and it is easy to imagine a smart mode for the emacs editor that will assist with basic syntax checking) the rules for correct

MathML are quite complicated and it is not really reasonable to expect average users to go to the bother of learning these rules. A comfortable tool is desirable, or a translator from more familiar formats.

This problem is less difficult to solve than the one of providing a general and seamless viewing method as discussed above. It is not necessary to agree on one generic solution and tastes may vary. Some preliminary tools already exits. The Amaya HTML editor [2] provides partial support for writing (and reading) MathML and the user interface is quite comfortable. However, Amaya is currently limited to a subset of MathML, basically the presentation tags. A comfortable user interface for the content part of MathML (where the appearance is not the whole story) is a challenging problem in user-interface design.

A tool called WebEQ [27] also supports MathML.

One way that a widely available commercial solution may come is that Mathematica and Maple appear to be intending to allow MathML export from their Mathematica notebook and Maple worksheet interfaces. At least for the community of users of these systems, that would appear to be a comfortable path to MathML and the input mechanisms of these systems already distinguish enough of the meaning of the formulae they manipulate to make it sensible to create content tagged MathML from these systems.

As the Euromath editor is based on similar technology to Amaya and Euromath has solved the problem of incporating multiple fonts for Mathematical symbols, it should be possible to build a more general mathematical editor by combining Euromath and Amaya. This is not yet done, however.

A TEX/LATEX to MathML equation converter is conceivable, but the rules for MathML content tags demand more information than can be found in generic TEX/LATEX formulae. Thus such a converter would need to make assumptions or to impose constraints on the input.

The IBM-Research product TechExplorer [26] implements an approach of this kind and should support MathML soon.

4 OpenMath

OpenMath [21, 22] began in late 1993 as an idea for a common language to be used as a way to exchange data between computer algebra (or symbolic computation) systems. After extensive discussions of the problems involved and the desirable scope of such a language OpenMath version 1.0 was agreed in late 1996.

The essential point of OpenMath is a method for encoding mathematical objects. OpenMath recognised that the number of different types of mathematical objects is huge and new types of objects are being invented by researchers on a regular basis. Hence a comprehensive catalogue of all types of mathematical

objects would be a list that would always be incomplete. Hence OpenMath has incorporated an extensible mechanism for listing the objects of interest within a particular well-defined field of Mathematics. This mechanism is known as a system of 'Content Dictionaries' and mathematical objects are encoded by the name of the appropriate Content Dictionary plus the classification of objects as specified in the dictionary. New content dictionaries can be created as the need arises to encode more types of mathematical objects. However, a symbolic or numerical system which wants to deal with OpenMath objects will need to have a special code for each of the Content Dictionaries that it can support. This code is known as a 'phrase book' which enables mappings from OpenMath objects using each Content Dictionary to the internal representation of the same objects in the system, and conversely from the system to OpenMath. Some systems such as typesetting or plotting systems might just read OpenMath objects and print or draw them and have no need of a converse mechanism to write in OpenMath.

The 'content' tags in MathML aim to solve the same sort of problem as OpenMath, but MathML has limited itself to a certain subset of Mathematics (relatively elementary Mathematics described roughly by the Mathematics encountered in courses on Mathematics from Kindergarten to Calculus and Linear Algebra). The aim of this part of MathML is to enable interactive mathematics to be carried out over the Web. OpenMath could serve the same purpose with a restricted set of Content Dictionaries and so there is a realm where OpenMath and MathML are competing technologies.

OpenMath and MathML are different in that the OpenMath design always assumed that special OpenMath code would have to be added to computer systems to make them speak OpenMath. Initially only Mathematical systems were being considered (and not Web browsers which would regard Mathematics as a small sectoral interest) and so this was not such an unreasonable approach. Moreover the question of typesetting or formatting mathematical objects was not of primary concern to OpenMath. Instead the question of capturing the mathematical meaning of objects was the main concern of OpenMath. For MathML to work, the question of displaying MathML on the Web is a major concern. OpenMath faces a similar problem as it is now planned to use OpenMath as a means of supplying interactive mathematical material (online books, electronic books on CD-ROM, distnace learning material with interactive mathematical formulae).

OpenMath was not concerned with any human typing in the OpenMath code for a formula, as it was always assumed that a symbolic computer system would be reading and writing OpenMath code. Superficially there is a similarity between OpenMath and MathML in that both use an SGML/XML format (though OpenMath has another more compact binary encoding as well). MathML aimed for an encoding that himans could digest.

However, despite these differences, it is not in the interest of OpenMath that

it should be completely incompatible with MathML in the areas of Mathematics where both could operate. An analysis of the approaches taken by MathML and OpenMath to the same areas of 'elementary' Mathematics has lead to a list of differences. Some are inessential differences or conflicts of nomenclature and some of these matters were resolved prior to the adoption of MathML. There are a number of other differences which are more fundamental and may be harder to resolve, but efforts are under way to do that. An ideal outcome would be that a formula written with the 'content' part of MathML could be understood as an OpenMath object by a foolproof process. A mechanism for embedding OpenMath objects in MathML (via extra commenst perhaps) is being considered.

MathML has the weight of the World Wide Web Consortium behind it, while OpenMath arose as a consensus between interested parties and has only an informal standing so far. On the other hand there has been a considerable effort in building OpenMath prototypes to test the validity of the theory, and it is intended to develop OpenMath in the light of experience with implementations. MathML seems less flexible.

In particular, this author believes that the style sheet problems mentioned above (section 3.4) are a serious obstacle to the success of the content part of MathML. Even the presentation part of MathML may not work properly unless the main browsers support the mathematical flow objects as explained in section 3.3. A satisfactory outcome for MathML would help OpenMath.

References

[1] Adobe Systems Incorporated, *Portable Document Format Reference Manual*, Adobe Systems Incorporated (1996).

[2] *Amaya*, URL `http://www.w3.org/Amaya/`.

[3] *AXIOM*, URL `http://www.nag.co.uk/symbolic/AX.html`.

[4] A. Capani, G. Niesi, L. Robbiano. *CoCoA, a system for doing Computations in Commutative Algebra*, 1995. URL `ftp://cocoa.dima.unige.it`.

[5] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, S. F. Smith, *Implementing Mathematics with the Nuprl Proof Development System*, Prentice Hall (1986). URL `http://simon.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html`.

[6] Projet Coq. *The Coq Proof Assistant*, version 6.2, URL `http://pauillac.inria.fr/coq/assis-eng.html`.

[7] John W. Eaton, *Octave*, Department of Chemical Engineering, University of Wisconsin-Madison,

URL http://bevo.che.wisc.edu/octave.

[8] *GAP*, URL ftp://ftp-gap.dcs.st-andrews.ac.uk/pub/gap.

[9] M. J. C. Gordon, T. F. Melham, *Introduction to HOL: A theorem proving environment for higher order logic*, Cambridge University Press (1993). URL http://www.cl.cam.ac.uk/Research/HVG/HOL/index.html.

[10] D. Grayson, M. Stillman, *Macaulay 2 home page*, URL http://www.math.uiuc.edu/Macaulay2/

[11] D. E. Knuth, *The TEXbook*, Addison-Wesley, Reading, Massachusetts (1992).

[12] L. Lamport, *LATEX: A Document Preparation System, (2nd ed.)*, Addison-Wesley, Reading, Massachusetts (1994).

[13] M. A. A. van Leeuwen, A. M. Cohen, B. Lisser, *LiE: A Package for Lie Group Computations*, CAN, Amsterdam (1992). URL http://www.can.nl/SystemsOverview/Special/GroupTheory/LiE/.

[14] *The LEGO Proof Assistant*. URL http://www.dcs.ed.ac.uk/home/lego.

[15] *Magma*. URL http://www.maths.usyd.edu.au:8000/u/magma/.

[16] *Waterloo Maple home page*, URL http://www.maplesoft.com.

[17] *Mathematical Markup Language (MathML) 1.0 Specification*, W3C Recommendation 07-April-1998. URL http://www.w3.org/TR/REC-MathML/.

[18] *MatLab*, URL http://matlab.mathworks.com/products/matlab/.

[19] R. Milner, M. Tofte, R. Harper, D. MacQueen, *The Definition of Standard ML (Revised)*, MIT Press, Cambridge MA (1997)

[20] The MuPAD Group (Benno Fuchssteiner et al.), *MuPAD User's Manual - MuPAD Version 1.2.2*. John Wiley and sons, Chichester, New York, first edition (1996). URL http://www.mupad.de.

[21] *OpenMath*, URL http://www.openmath.org/.

[22] *OpenMath: Accessing and Using Mathematical Information Electronically*, ESPRIT project 24.969, URL http://www.nag.co.uk/projects/OpenMath.html.

[23] *Pari*, URL ftp://megrez.ceremab.u-bordeaux.fr/pub/pari/.

[24] *REDUCE*, URL http://ftp.rand.org/software_and_data/reduce.

[25] *SciLab*, URL ftp://ftp.inria.fr/INRIA/Projects/Meta2/Scilab/.

[26] *TechExplorer*, URL http://www.alphaworks.ibm.com/formula/techexplorer.

[27] *WebEQ*, URL http://www.webeq.com/.

[28] S. Wolfram, *The Mathematica Book: Third Edition, Mathematica Version 3*, Wolfram Research, Inc. 1996. URL http://www.mathematica.com/

[29] *Extensible Markup Language (XML) 1.0*, W3C Recommendation 10-February-1998, URL http://www.w3.org/TR/1998/REC-xml-19980210.

[30] *A Proposal for XSL*, Submitted to W3C on 27 August 1997, URL http://www.w3.org/TR/NOTE-XSL.html.

School of Mathematics
Trinity College
Dublin 2
Ireland

Email: richardt@maths.tcd.ie

This paper was presented at the workshop “Electronic Publishing Systems for Science and Education” held at Pereslavl-Zalessky, Russia, May 12-14, 1998.