Date: November 2002

OpenMath – Guidelines for Tool Developers

Stephen Buswell[1], James Davenport[2], David Carlisle[3] and Mike Dewar[3]

[1]Stilo [2]Univ.Bath [3]NAG

© 2002 The OpenMath Thematic Network

Contents

1 Introduction

1.1 Purpose of this Document

The intention of this document is to provide a starting point for developers creating new OpenMath applications or adding OpenMath functionality to existing applications. It gives the user a high-level view of some common tasks, and some issues which need to be considered when building an OpenMath-aware system. It also indicates where many useful resources (both information and software) may be found.

1.2 OpenMath Overview

OpenMath is a standard for the exchange of semantically rich mathematical objects between communicating applications. The standard and other related documents and resources can be found at `http://monet.nag.co.uk/cocoon/openmath`. A useful introductory tour can be found at: `http://www.risc.uni-linz.ac.at/~ocaprott/pisa2002/index.html`.

OpenMath structures mathematics into Content Dictionaries (CDs) which contain symbols with precisely defined mathematical semantics. Two applications can communicate if they both implement the OpenMath semantics of a given CD, or an agreed subset of that CD. For ease of communication and negotiation, CDs may be grouped into CD Groups. The creation and use of CDs is discussed in more detail below.

OpenMath defines an abstract model for constructing mathematical objects from combinations of these symbols, together with two concrete interchange formats, one XML-based, one binary.

The conversion of an OpenMath object between an interchange representation and the internal representation in a software application is performed by an interface program called a Phrasebook. The translation is governed by the Content Dictionaries and the specifics of the application.

1.3 OpenMath and MathML

OpenMath has a close relationship with MathML, the W3C recommendation for mathematics on the web, `http://www.w3.org/Math`. MathML has support for both presentation and content models of mathematical expressions. The presentation side is frequently used to provide a rendering mechanism for OpenMath, which has no native rendering scheme. Conversely, OpenMath can be used as an encoding mechanism to extend Content MathML beyond its basic scope.

Where OpenMath and MathML overlap, i.e., OpenMath CDs exist containing symbols which also have MathML content elements assigned, the semantics of the symbols in the two representations have been aligned. An OpenMath CD Group, the 'MathML Compatibility Group' has been defined, containing the OpenMath forms of exactly those symbols which occur in Content MathML.

2 Common Tasks

2.1 Rendering

OpenMath editing tools (see 2.3) provide native OpenMath rendering facilities, however for most applications rendering of OpenMath applications will be via translation of the semantically oriented OpenMath object to a different form more directly associated with a rendering technology.

LaTeX and (Presentation) MathML are two obvious possibilities for specifying mathematical rendering, however other possibilities also exist. For example most computer algebra systems have methods of rendering mathematical expressions which are encoded in the language of that system, thus a *phrasebook* which maps OpenMath objects to a system such as Mathematica or Maple may be used to obtain a rendering of the object in addition to its use to encode the semantics of the expression in the language of the computational system. It may be possible to have a phrasebook that is specifically tied to rendering that may be used for a wider class of OpenMath objects as for a rendering-specific phrasebook there is less need to implement the detailed mathematical functionality expressed in the Content Dictionary in the Computer Algebra system being used.

2.1.1 Rendering via TeX

For rendering via TeX one needs to use a stylesheet in some language to convert to TeX mathematical markup, from which dvi, postscript, pdf and other formats may be easily derived. The stylesheet should ideally be easily extensible (see 3.1) and allow presentation rules for symbols introduced in new Content Dictionaries to be easily added. It is likely that any such translation will require XML encoded OpenMath as input, the binary OpenMath encoding may be supported by the use of an OpenMath Application that reads the binary input and outputs the object in the XML encoding.

One such stylesheet implementations is available as part of the OMDoc project: `http://www.mathweb.org/omdoc/`.

An alternative route to TeX rendering is to first translate to Presentation MathML as described below, and then to use a stylesheet that converts Presentation MathML to TeX.

`http://monet.nag.co.uk/cocoon/openmath/software/mml-files/index.html` DSSSL stylesheet for MathML Render MathML to TeX or RTF (For Microsoft Word)

`http://www.raleigh.ru/MathML/mmltex/index.php?lang=en` Vasil Yaroshevich's MathML to LaTeX stylesheets.

2.1.2 Rendering via MathML

There are now several available stylesheets aimed at converting OpenMath to MathML, for example:

`http://www.mathweb.org/omdoc/pres-architecture.html` OpenMath - MathML stylesheets from the OMDoc Project.

`http://www.cs.nmsu.edu/~bpalmer/cmml2om` OpenMath - MathML stylesheets from Brian Palmer.

Once the MathML is encoded in presentation MathML, it may be viewed in a stand alone MathML viewer (such as the HELM project's viewer, `http://helm.cs.unibo.it/mml-widget`) or more commonly via a MathML enabled web browser. The current state of MathML rendering in browsers is tabulated at the W3C's MathML pages `http://www.w3.org/Math/XSL`, but in brief, Presentation MathML rendering is supported in Amaya, Mozilla and (from Version 7) Netscape browsers. For Internet Explorer free (no cost) "Behavior" extension is available from Design Science `http://www.dessci.com/webmath/mathplayer` which will render both Content and Presentation MathML. There are some differences in the way MathML is supported in these browsers, principally the lack of support for Content MathML in Mozilla/Netscape implementation and the fact that using the Behaviors interface in Internet Explorer requires some non standard Object elements and Processing Instructions at the head of the document. However a stylesheet is available `http://www.w3.org/Math/XSL` which detects the client being used and modifies the document accordingly. This allows XHTML+MathML documents to be served and displayed without change on Amaya, Mozilla, Netscape and Internet Explorer. By default, Internet Explorer's security settings do not allow XHTML pages on one server to access stylesheets on a different server, so it is recommended that authors take a copy of this stylesheet and reference a local copy of the stylesheet from their pages.

2.1.3 Rendering via SVG

A third option for rendering OpenMath would be to convert to the W3C's Scalable Vector Graphics Language (SVG). As for TEX, one could write stylesheets directly mapping to SVG but a common idiom is to first map to Presentation MathML. Using MathML as an intermediate format in this way allows the layout rules for the symbols in a Content Dictionary to be specified just once, as Presentation MathML, and then support for rendering objects using that Dictionary is automatically obtained for any rendering technology for which you have MathML support.

One product implementing MathML to SVG conversion is available from SchemaSoft: `http://www.schemasoft.com/MathML` .

OpenMath to SVG is also implemented in Nice: `http://mainline.essi.fr`.

2.2 Phrasebooks

While it is possible to write a phrasebook from scratch, there are a number of tools and components to make the task easier. A phrasebook has to do one or both of the following:

1. parse an OpenMath object and translate it to an appropriate internal (native) representation;
2. translate a native object into OpenMath.

The process of translation requires both an understanding of the semantics of a set of content dictionaries, and the semantics of the native data structures.

There are a number of tools to help build phrasebooks. The INRIA libraries (in C, C++ and Java) provide, amongst other things, an API which reads an OpenMath object from a stream or file and returns a series of tokens. These tokens may then be interpreted by a user's code and

built up into a suitable native object. The latest versions of the libraries are available from the OpenMath software page at `http://monet.nag.co.uk/openmath/software`.

There is also a Java library from RIACA which provides a higher-level, more object-oriented API in terms of the abstract OpenMath objects defined in the OpenMath Standard. So whereas the INRIA libraries read an OpenMath encoding and return a series of tokens, the RIACA library will return an instance of an OpenMath object. This is often easier to use from a programming point of view, but for large objects it is less efficient since both the OpenMath and native representations must be created in memory. The library, along with examples of phrasebooks constructed for both GAP and Mathematica, is available from the RIACA OpenMath page at `http://www.riaca.win.tue.nl`.

A related tool from RIACA is their CD Editor. While primarily a tool for creating and editing content dictionaries, it will generate a Java Codec for a phrasebook which is compatible with the RIACA library, so making it easier to add support for new CDs to a phrasebook. At the time of writing a beta version of the editor is available from `http://www.riaca.win.tue.nl/download/om/cd/editor`.

Ad-hoc phrasebooks have also been developed based on the XSLT stylesheets described elsewhere, in particular for applications which support the reading and/or writing of Content MathML. This is only really a practical approach for phrasebooks designed to handle a small number of content dictionaries.

2.3 Editing and Manipulating OpenMath objects

OpenMath is a standard for the interchange of mathematical objects between communicating applications, so in many applications there is no direct interface between the user and the OpenMath objects. However in some cases it is useful to create objects separately. A number of OpenMath-aware tools and components exist.

2.3.1 OpenMath Editing Tools

`http://mainline.essi.fr/jome/jome-en.html` The JOME OpenMath Editor. A Java component. It supports graphical display and editing of mathematical expressions.

`http://aiki.ccaps.cs.cmu.edu/DownloadIndex.html` OMDoc emacs mode for writing OMDoc XML documents. OpenMath DTD-aware, with element-completion and context-sensitive menus.

`mailto:sb@stilo.com` STARS a MathML / OpenMath editor applet which can be integrated into other systems. STARS uses the Webeq MathML rendering applet to render mathematics in web pages. It takes a TeX-like linear syntax for mathematical input and translates this into MathML and OpenMath. Input can also be given in MathML and OpenMath. It has been used as a front-end to various OpenMath systems, including COQ and the NAG multiple-integrator demonstrator.

`http://www.matracas.org` QMath, a tool to produce OpenMath and OMDoc with a text-based input syntax

`http://www.maths.tcd.ie/~richardt/openmath/` LaTeX to OpenMath translator demo

Another possibility is to use a MathML-aware Computer Algebra application such as Maple

(`www.maplesoft.com`) export the MathML and then use one of the MathML-OpenMath translators, This conversion is likely to be more successful if the Computer Algebra system exports Content mathML rather than presentation.

2.4 Integrating Mathematical Applications

OpenMath was originally designed as a data exchange format for mathematical applications, however it does not prescribe any particular transport mechanism which should be used to move OpenMath object from one system to another. There is a growing list of environments which do this, an up-to-date list of which can be found on the OpenMath software page at `http://monet.nag.co.uk/openmath/software`. We describe a number of them briefly here.

JavaMath (`http://javamath.sourceforge.net`) is an easy-to-use suite of software which enables existing computational engines to be accessed from Java programs. It works by wrapping the engine in a server layer so that it can, in principle, be accessed from anywhere on the internet. All data is passed using OpenMath.

RIACA have a number of components for deploying computational engines as servers, and a *shell* which allows a variety of such engines to be accessed from a single environment. They are all available from `http://www.riaca.win.tue.nl/products/index.html`.

A number of more comprehensive software environments for allowing mathematical systems to interact, using OpenMath as a data representation language, are under development. These include:

- *OpenXM* (`http://www.math.sci.kobe-u.ac.jp/OpenXM`)
- *IAMC* (`http://icm.mcs.kent.edu/research/iamc`)
- *MathWeb* (`http://www.mathweb.org`)
- *LogicBroker* (`http://www.mrg.dist.unige.it/~nembo/project.htm`)

3 Issues

3.1 Extensibility

OpenMath is inherently extensible, there is a small core set of features, Application, Binding, Integers, Floating Point numbers, etc., but the majority of the semantics of any OpenMath expression is specified by reference to one or more Content Dictionaries. It is the intention that these dictionaries should be created as needed to encode the semantics of different areas of mathematics.

This extensibility means that if possible generic OpenMath tools should be able to handle new Content Dictionaries, either automatically (for example an OpenMath Validator might not require anything other than the list of symbol names which it can extract directly from the CD file) or by explicit parameterisation, for example a stylesheet converting OpenMath to MathML might require additional stylesheet modules specifying the rendering of each CD, but it should be possible to easily incorporate all the modules required for any particular OpenMath Object.

Of course some OpenMath applications are not intended to be general and will explicitly declare the Content Dictionaries that they can handle. This will be the usual case for applications that

implement specific mathematical functionality. Even in these cases the application should accept objects using any CD, but as explained in the OpenMath standard, may use the standard error CD to report errors for objects that it can not process.

3.2 The XML and Binary OpenMath Encodings

The OpenMath standard provides two concrete representations for OpenMath objects: one binary and one XML-based. Standard libraries and phrasebooks exist supporting each representation. The decision of which to use is application-specific.

In general one can say that the binary encoding is more efficient to store and transmit.

The XML encoding is more verbose, although some improvement for transmission can be obtained using schemes such as gzip as OpenMath XML is text-based and highly-repetitive. XML is more compatible with industry standard tools such as XSLT (see also below). It is also compatible with emerging XML-based interface and protocol standards such as WSDL (Web Services Definition Language) and SOAP (Simple Object Access Protocol) which may provide a basis for mathematical services on the web.

3.3 Using Generic XML tools with OpenMath

The OpenMath XML encoding is a restricted subset of XML, so any valid OpenMath object is a well-formed XML document, and can be validated using the OpenMath DTD and any validating XML parser. A non-normative XSD Schema for OpenMath objects can be found on the OpenMath site and can be used with standard schema-aware systems. It should be borne in mind that the DTD is fairly permissive, and validation at the XML level does not guarantee that the object is a valid OpenMath object according to the other conditions imposed by the standard.

As has been noted above under Rendering, this XML nature can be exploited for rendering OpenMath objects in a browser using XSLT stylesheets. XSLT transformations can also be used to translate from the OpenMath encoding to simple text-based input formats for some mathematical applications.

Generic XML editors can also be used to create OpenMath objects, although the internal structure of an OpenMath object does not make it very amenable to meaningful display in document-oriented systems.

3.4 The Semantics of OpenMath Applications

OpenMath applications will vary in the precise mathematical semantics that they possess, and also in the level at which they process mathematics. Some applications may be computer algebra systems or theorem provers, others will be databases or text handling applications. For the former class of applications, the question of which OpenMath constructs they understand is closely linked to their internal semantics. For the second, there is an interesting question of how they cope with the extensibility of OpenMath: new Content Dictionaries can be written at any time, and the application should be able to grow with this. One solution is to build a parallel dictionary to the Content Dictionary structure, so that an application that translated OpenMath into ZZZ would have a family of files with names such as `arith1.zzz`, which would contain, in

a format known to the application, the rules for converting the symbols in the `arith1` Content Dictionary into ZZZ. While this is not the only way of being extensible, experience has shown that hard-coding Content Dictionaries into converters leaves major maintenance problems.

3.4.1 Understanding and Using OpenMath Content Dictionaries (CDs)

> Content Dictionaries are used to assign informal and formal semantics to all symbols arising in OpenMath objects. ... The application receiving the object may then recognize whether or not, according to the semantics of the symbols defined in the Content Dictionaries, the object can be transformed to the corresponding internal representation used by the application. [7]

A few additional remarks need to be made.

- The first is that it is not so much a question of an "application" understanding a Content Dictionary, rather it is a case of the application and an associated phrasebook understanding the Content Dictionary. For example, it would be totally legitimate for a Pascal generator to claim to understand the `transc1` Content Dictionary, even though the only trigonometric functions defined in Pascal [4] are sin, cos and arctan, and no hyperbolic functions are defined, since the translator could convert hyperbolic functions in terms of exp, their inverses in terms of ln, and missing trigonometric functions in terms of the existing ones, though special cases would have to be coped with, as in the OpenMath $\operatorname{arccot} z$, which could be translated into Pascal as $\arctan \frac{1}{z}$, or probably needs a more complicated construct such as

  if $z = 0$ **then** $\frac{\pi}{2}$ **else** $\arctan \frac{1}{z}$.

- An application can only be expected to understand a Content Dictionary within the application's own intrinsic limitations. Thus, although

  ```
  <OMS cd="arith1" name="plus"/>
  ```

  is defined to be associative, a fixed-precision numerical processor cannot guarantee to observe this.

- A key idea is that an application/phrasebook combination must not mis-understand a symbol in a Content Dictionary. For example, for the reasons stated in section 3.4.2.2.2, a Maple phrasebook which translated the OpenMath arccot into the Maple one would definitely have misunderstood that symbol.

3.4.2 Private or Public Content Dictionaries

Content Dictionaries are the main means by which OpenMath gains its unique combination of extensibility and interoperability. This means that most application designers will be forced to face the question:

> which Content Dictionaries should I use? In particular, what should be the balance between public Content Dictionaries (from `www.openmath.org`) and private ones that I invent myself? If I use private ones, should I keep them private, or submit them to the OpenMath society via `http://monet.nag.co.uk/openmath/cdfiles/contrib`?

There is also an interesting half-way house, of using the public CD, but attaching a private annotation, using the `OMATTR` construct [7].

It is difficult to lay down an absolute decision process for this, as the issues depend both on the symbols one wishes to use and on the intent of the application. Instead, we will state some general rules, and illustrate these issues by a variety of examples from "definitely public" to "definitely private".

3.4.2.1 Definitely Public If the semantics of the concept you want are the same as the semantics of an existing symbol in a public CD, then one should clearly use that symbol. So, for example, the function $\sin : \mathbb{C} \to \mathbb{C}$ (see section 3.4.2.2.1 for more complicated signatures) should definitely be encoded by

```
<OMS cd="transc1" name="sin"/>
```

and it would be bad for interoperability to use anything else.

3.4.2.1.1 `arith1` or `arith2` One special case of the above rule needs considering. Both the `arith1` and `arith2` CDs contain the symbol `times`. The difference is that the one in `arith2` is definitely declared to be commutative. There is one general principle, but it has to be tempered with "fitness for purpose" considerations.

If it would be wrong to apply the operation to non-commutative objects, then one should use

```
<OMS cd="arith2" name="times"/>
```

For example, a linear algebra system that implements Gaussian elimination should, in principle, insist on `arith2`, since Gaussian elimination does not work on non-commutative objects. However, if the system is not polymorphic, and only operates on, say, IEEE floating-point numbers [5], then these are known to be commutative, so it would be sane for it to accept `arith1`'s version of `times`. This is an example of what we mean by "fitness for purpose" considerations.

3.4.2.1.2 `transc1` or `transc3` Here, by contrast, the choice is much clearer. The functions defined in `transc3` are multi-valued analogues of the functions in the `transc1` Content Dictionary, and therefore have different signatures. Most of them therefore return infinite sets, e.g. $\underbrace{\arctan}_{\texttt{transc3}}(0) = \{n\pi \mid n \in \mathbf{Z}\}$. It is therefore unlikely that a general-purpose computer algebra system would use these in their standard interface.

3.4.2.2 The semantics are not quite right This can occur in several ways.

3.4.2.2.1 The signature is wrong One example of this would be the use of sin, defined by the usual power series $\sin x = x - \frac{1}{6}x^3 + \frac{1}{24}x^5 + \cdots$, but acting on, and returning, square matrices. This is definitely not within the scope of the signature from `transc1.sts` [3] for

```
<OMS cd="transc1" name="sin"/>
```

There are essentially three possibilities.

1. Ignore the issue, and use

   ```
   <OMS cd="transc1" name="sin"/>
   ```

2. Use the public symbols, but add a private attribute to keep track of the difference, as in

   ```
   <OMATTR>
     <OMATP>
       <OMS cd="mytypes" name="type"/>
       <OMS cd="mytypes" name="matrixvalued"/>
     </OMATP>
     <OMS cd="transc1" name="sin"/>
   </OMATTR>
   ```

   By the standard rules on attributes, other systems that did not know about the `mytypes` CD could treat this as in the previous case.

3. Use a completely different CD, as in

   ```
   <OMS cd="matrixtransc" name="sin"/>
   ```

Which to use depends on the context and the rôle of the application. We give a few examples.

- An electronic book on matrices. If there is no computational support for the chapter on elementary functions of matrices, then it would be sane to use option 1, though if the OpenMath is going to be searchable, then option 2 might be better, to distinguish this use of sin from the $\mathbb{C} \to \mathbb{C}$ one.
- A dedicated system for algebra and arithmetic with elementary functions on matrices. Since no-one would be calling this system unless they already knew that elementary functions on matrices was the name of the game, option 3 would seem to be the best.

3.4.2.2.2 The semantics are subtly different We will illustrate this with two examples from the inverse elementary functions: see [2].

arccot The branch cuts for arccot have been a subject of some controversy: see the table 3.4.2.2.2. OpenMath uses the definition in [1, ninth printing]. This clearly poses problems for computer algebra systems such as Maple and Axiom. The solution for the phrasebook for such systems advocated in [2] is as follows.

OpenMath→System The OpenMath arccot z should be translated into Maple as $\arctan \frac{1}{z}$, or possibly a more complicated construct such as

if $z = 0$ **then** $\frac{\pi}{2}$ **else** $\arctan \frac{1}{z}$.

System→OpenMath This is distinctly more difficult, if the system has generated some arccot expressions itself (it should never see them on input, of course). The best translation is $\operatorname{arccot} z \mapsto \frac{\pi}{2} - \arctan z$.

Table 1: Different values of arccot(−1)

This table is taken from [2], with the addition of the Maxima line.

Source	Detail	arccot(−1)	Comments
[1]	1st printing	$3\pi/4$	inconsistent
[1]	9th printing	$-\pi/4$	
[6]	5th edition	?	inconsistent
[10]	30th edition	$3\pi/4$	inconsistent
Maple	V release 5	$3\pi/4$	
Axiom	2.1	$3\pi/4$	
Mathematica	[9]	$-\pi/4$	
Maxima	5.5	$-\pi/4$	
Reduce	3.4.1	$-\pi/4$	in floating point
Matlab	5.3.0	$-\pi/4$	in floating point
Matlab	5.3.0	$3\pi/4$	symbolic toolbox

The note "inconsistent" means that, although the source quotes, or clearly lets be inferred, a value for arccot(−1), there are enough inconsistencies in the definition of arccot that one could infer a different value. For [6], $3\pi/4$ and $-\pi/4$ are equally inferable.

arctan Derive's definition of arctan is subtly different from that of [1]: differing only on the branch cuts. The definition of [1] has the following relation with arcsin:

$$\arcsin z = \arctan \frac{z}{\sqrt{1-z^2}} + \pi\mathcal{K}(-\ln(1+z)) - \pi\mathcal{K}(-\ln(1-z)), \tag{1}$$

where $\mathcal{K}$ denotes the unwinding number [2]. Derive has a different definition of arctan to eliminate the unwinding numbers from equation (1), so that, for Derive, $\arcsin(z) = \underbrace{\arctan}_{\text{Derive}} \frac{z}{\sqrt{1-z^2}}$. This definition can be related to that of OpenMath via

$$\underbrace{\arctan}_{\text{Derive}}(z) = \overline{\arctan \overline{z}}. \tag{2}$$

This representation has the advantage that the rule $\overline{\overline{z}} \to z$ means that the transformation is self-inverse. Hence we could argue for the following transformations.

OpenMath→Derive Apply equation (2) to every occurrence of OpenMath's arctan.
Derive→OpenMath Apply equation (2) to every occurrence of Derive's arctan.

However, if we are only using Derive as an integrator over $\mathbb{R}$, and if we know that it will only return real arctans (rather than canceling complex ones) then we need not do any transformations, since arctan and $\underbrace{\arctan}_{\text{Derive}}$ agree as (partial) functions $\mathbb{R} \to \mathbb{R}$.

3.4.2.2.3 The semantics are very different One example of this is the concept of approximate gcd of polynomials. The ordinary symbol

```
<OMS cd="poly" name="gcd"/>
```

is defined to be `nassoc` [3], which means that it takes an arbitrary number of arguments, and is associative on them, so that

$$\gcd(\gcd(a,b),c) = \gcd(a,b,c) = \gcd(a,\gcd(b,c)). \tag{3}$$

While there is (now) general consensus on the overall shape of a definition of $\epsilon - \gcd$, *viz.* that an $\epsilon - \gcd$ of $p_1, \ldots, p_n$ is a polynomial q of highest degree such that there exist polynomials $\delta_1, \ldots, \delta_n$ of size less than ϵ such that

$$q = \gcd(p_1 + \delta_1, \ldots, p_n + \delta_n),$$

there is less consensus on on the details, in particular the definition of size, which can be, for $\delta = a_k x^k + \cdots a_0$, any of:

$||\cdot||_\infty$: $\max(|a_0|, \ldots, |a_k|)$;
$||\cdot||_2$: $\sqrt{\sum_{i=0}^{k} a_i^2}$;
$||\cdot||_1$: $\sum_{i=0}^{k} |a_i|$.

While there are in principle the same three choices of symbol as in section 3.4.2.2.1, the choice is much simpler. Option 1 is definitely wrong: there is no way of specifying ϵ, and the semantics are totally wrong: the operation of $\epsilon - \gcd$ is not associative, and in some formulations not commutative [8]. Option 2 could solve the first problem, in a formulation such as

```
<OMATTR>
  <OMATP>
    <OMS cd="mytolerance" name="L2Norm"/>
    <OMA>
      <OMS cd="arith1" name="power"/>
      <OMI> 10 </OMI>
      <OMI> -6 </OMI>
    </OMA>
  </OMATP>
  <OMS cd="poly" name="gcd"/>
</OMATTR>
```

but it does not solve the more fundamental problem. Again, in the "electronic book" context described in subsection 3.4.2.2.1, one could argue for option 2, but here there is the disadvantage that a renderer might not know about the `mytolerance` CD, and would therefore drop the tolerance, so that the rendered result would appear indistinguishable from an ordinary gcd. However, the thesis of these guidelines is embodied in the following rule, which strongly suggests option 3.

If the mathematical and syntactic properties such as commutativity and associativity defined in the formal mathematical properties and STS file no longer hold for the public symbol in the new context, then a new symbol must be used.

4 Conclusion

This document surveys some of the issues relating to building an OpenMath application, and has links to several example applications. A more extensive list, which is updated as new software is announced is available from the OpenMath web site at `http://monet.nag.co.uk/openmath/software`. This page should be consulted for any updates to the software mentioned in this document.

If you do produce new OpenMath software it may be announced on the public mailing list `om-announce@openmath.org`. Information on subscribing to this mailing list is also available from the OpenMath web site `http://monet.nag.co.uk/openmath/lists`.

5 References

[1] Abramowitz,M. & Stegun,I., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. US Government Printing Office, 1964. 10th Printing December 1972.

[2] Corless,R.M., Davenport,J.H., Jeffrey,D.J. & Watt,S.M., "According to Abramowitz and Stegun". *SIGSAM Bulletin* **34** (2000) 2, pp. 58–65.

[3] Davenport,J.H., A Small OpenMath Type System. *ACM SIGSAM Bulletin* **34** (2000) 2 pp. 16–21.

[4] IEEE Standard Pascal Computer Programming Language. IEEE Inc., 1983.

[5] IEEE Standard 754 for Binary Floating-Point Arithmetic. IEEE Inc., 1985. Reprinted in *ACM SIGPLAN Notices* **22** (1987) pp. 9–25. See also `http://www.cs.berkeley.edu/~wkahan/ieee754status/754story.html`.

[6] Gradshteyn,I.S. & Ryzhik,I.M. (ed A. Jeffrey), Table of Integrals, Series and Products. 5th ed., Academic Press, 1994.

[7] Caprotti,O., Carlisle,D.P. & Cohen,A.M., The OpenMath standard. The OpenMath Consortium, 2000. `monet.nag.co.uk/openmath/standard`.

[8] Rupprecht,D., *Éléments de Géométrie algébrique approchée : Étude du PGCD et de la factorisation.* PhD thesis, Université de Nice–Sophia Antipolis, Jan. 2000.

[9] Wolfram,S., *The Mathematica Book*. Wolfram Media/C.U.P., 1999.

[10] Zwillinger,D. (ed.), *CRC Standard Mathematical Tables and Formulae*. 30th. ed., CRC Press, Boca Raton, 1996.

[11] S. Dalmas, M. Gaëtano and S. Watt, *An OpenMath 1.0 Implementation*. Proceedings of ISSAC 97, ACM Press 1997.