The *OpenMath* Standard

Version: 2.0 Public Draft 1 (13 October 2003)
The OpenMath Society
Editors
S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, M. Kohlhase
October 2003

Abstract

This document proposes *OpenMath* as a standard for the communication of semantically rich mathematical objects. This draft of the *OpenMath* standard comprises the following: a description of *OpenMath* objects, the grammar of xml and of the binary encoding of objects, a description of Content Dictionaries and an xml document type definition for validating Content Dictionaries. The non-normative Chapter 1 of this document briefly overviews the history of *OpenMath*.

Change-marked edition notes

This edition contains colour coded change markings relative to the OpenMath 1.0 document...

- New text is marked with css class "new" (green).
- ~~Deleted text is marked with css class "del" (red).~~

Sections with new text

3.1 Formal Definition of OpenMath Objects
3.1.1 Basic OpenMath objects
3.1.2 Derived OpenMath Objects
3.1.3 Compound OpenMath Objects
3.1.4 OpenMath Symbol Rôles
3.2 Further Description of OpenMath Objects
3.3 Names
3.4 Summary
4.1.1 A GrammarSchema for the xml Encoding
4.1.2 Informal description of the Grammarxml Encoding
4.1.2.1 An Acyclicity Constraint
4.1.2.2 Sharing and Bound Variables
4.1.3 Embedding OpenMath in xml Documents
4.2 The Binary Encoding
4.2.1 A Grammar for the Binary Encoding
4.2.2 Description of the Grammar
4.2.2.1 Sharing in Objects begining with the identifier [24]
4.2.3 Sharing with References (beginning with [24+64])
4.2.4 Implementation Note
4.2.6 Relation to the OpenMath1 binary encoding
5.1 Introduction
5.2 Abstract Content Dictionaries
5.2.1 Content Dictionary Status
5.2.2 Content Dictionary Version Numbers
5.3 The xmlReference Encoding for Content Dictionaries

Contents

List of Figures

Chapter 1
OpenMath Movement

This chapter is a historical account of *OpenMath* and should be regarded as non-normative.

OpenMath is a standard for representing mathematical objects, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. While the original designers were mainly developers of computer algebra systems, it is now attracting interest from other areas of scientific computation and from many publishers of electronic documents with a significant mathematical content. There is a strong relationship to the MathML recommendation [19] from the Worldwide Web Consortium, and a large overlap between the two developer communities. MathML deals principally with the *presentation* of mathematical objects, while *OpenMath* is solely concerned with their semantic meaning or *content*. While MathML does have some limited facilities for dealing with content, it also allows semantic information encoded in *OpenMath* to be embedded inside a MathML structure. Thus the two technologies may be seen as highly complementary.

1.1 History

OpenMath was originally developed through a series of workshops held in Zurich (1993 and 1996), Oxford (1994), Amsterdam (1995), Copenhagen (1995), Bath (1996), Dublin (1996), Nice (1997), Yorktown Heights (1997), Berlin (1998), and Tallahassee (1998). The participants in these workshops formed a global *OpenMath* community which was coordinated by a Steering Committee and operated through electronic mailing groups and ad -hoc working parties. This loose arrangement has been formalised through the establishment of an *OpenMath* Society. Up until the end of 1996 much of the work of the community was funded through a grant from the Human Capital and Mobility program of the European Union, the contributions of several institutions and individuals. A document outlining the objectives and basic design of *OpenMath* was produced (later published as [1]). By the end of 1996 a simplified specification had been agreed on and some prototype implementations have come about [5].

In 1996 a group of European participants in *OpenMath* decided to bid for funding under the European Union's Fourth Framework Programme for strategic research in information technology. This bid was successful and the project started in late 1997. The principal aims of the project are to formalise *OpenMath* as a standard and to develop it further through industrial applications; this document is a product of that process and draws heavily on the previous work described earlier. *OpenMath* participants from all over the world continue to meet regularly and cooperate on areas of mutual interest, and recent workshops in Tallahassee (November 1998) and Eindhoven (June 1999) endorsed drafts of this document as the current *OpenMath* standard.

1.2 *OpenMath* Society

In November 1998 the *OpenMath* Society has been established to coordinate all *OpenMath* activities. The society is based in Helsinki, Finland and is steered by the executive committee whose members are elected by the society. The official web page of the society is http://www.openmath.org.

Chapter 2
Introduction to *OpenMath*

This chapter briefly introduces *OpenMath* concepts and notions that are referred to in the rest of this document.

2.1 *OpenMath* Architecture
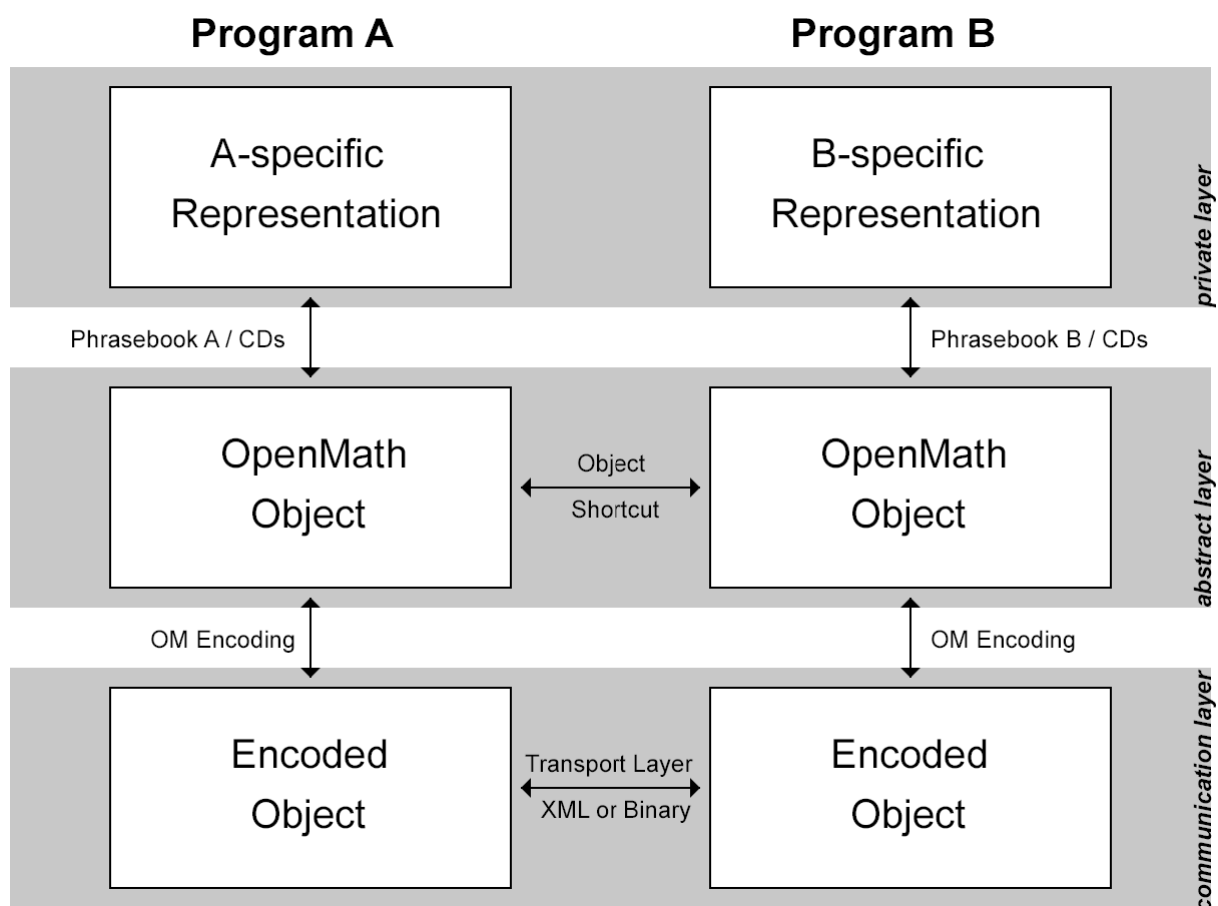
Figure 2.1 The *OpenMath* Architecture

The architecture of *OpenMath* is described in Figure 2.1 and summarizes the interactions among the different *OpenMath* components. There are three layers of representation of a mathematical object [13]. A private layer that is the internal representation used by an application. An abstract layer that is the representation as an *OpenMath* object. Third is a communication layer that translates the *OpenMath* object representation to a stream of bytes. An application dependent program manipulates the mathematical objects using its internal representation, it can convert them to *OpenMath* objects and communicate them by using the byte stream representation of *OpenMath* objects.

2.2 *OpenMath* Objects and Encodings

OpenMath objects are representations of mathematical entities that can be communicated among various software applications in a meaningful way, that is, preserving their "semantics".

OpenMath objects and encodings are described in detail in Chapter 3 and Chapter 4.

The standard endorses encodings in xml and binary format. These are the encodings supported by the official *OpenMath* libraries. However they are not the only possible encodings of *OpenMath* objects. Users that wish to define their own encoding using some other specific language (e.g. Lisp) may do so provided there is an effective translation of this encoding to an official one.

2.3 Content Dictionaries

Content Dictionaries (CDs) are used to assign informal and formal semantics to all symbols used in the *OpenMath* objects. They define the symbols used to represent concepts arising in a particular

area of mathematics.

The Content Dictionaries are public, they represent the actual common knowledge among *OpenMath* applications. Content Dictionaries fix the "meaning" of objects independently of the application. 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.

2.4 Additional Files

Several additional files are related to Content Dictionaries. Signature files contain the signatures of symbols defined in some *OpenMath* Content Dictionary and their format is endorsed by this standard.

Furthermore, the standard fixes how to define as a CDGroup a specific set of Content Dictionaries.

Auxiliary files that define presentation and rendering or that are used for manipulating and processing Content Dictionaries are not discussed by the standard.

2.5 Phrasebooks

The conversion of an *OpenMath* object to/from the internal representation in a software application is performed by an interface program called *Phrasebook*. The translation is governed by the Content Dictionaries and the specifics of the application. It is envisioned that a software application dealing with a specific area of mathematics declares which Content Dictionaries it understands. As a consequence, it is expected that the Phrasebook of the application is able to translate *OpenMath* objects built using symbols from these Content Dictionaries to/from the internal mathematical objects of the application.

OpenMath objects do not specify any computational behaviour, they merely represent mathematical expressions. Part of the *OpenMath* philosophy is to leave it to the application to decide what it does with an object once it has received it. *OpenMath* is not a query or programming language. Because of this, *OpenMath* does not prescribe a way of forcing "evaluation" or "simplification" of objects like $2 + 3$ or $\sin(\pi)$. Thus, the same object $2 + 3$ could be transformed to 5 by a computer algebra system, or displayed as $2 + 3$ by a typesetting tool.

Chapter 3
OpenMath Objects

In this chapter we provide a self-contained description of *OpenMath* objects. We first do so at an informal level (Section 3.2) and next by means of an abstract grammar description (Section 3.1).

3.1 Formal Definition of *OpenMath* Objects

OpenMath represents mathematical objects as terms or as labelled trees that are called *OpenMath* objects or *OpenMath* expressions. The definition of an abstract *OpenMath* object is then the following.

3.1.1 Basic *OpenMath* objects

The Basic *OpenMath* Objects form the leaves of the *OpenMath* Object tree. A Basic *OpenMath* Object is of one of the following.

- (i) Integer.

  Integers in the mathematical sense, with no predefined range. They are "infinite precision" integers (also called "bignums" in computer algebra).

- (ii) IEEE floating point number.

  Double precision floating-point numbers following the ieee 754-1985 standard [7].

- (iii) Character string.

  A Unicode Character string. This also corresponds to `characters' in xml.

- (iv) Bytearray.

  A sequence of bytes.

- (v) Symbol.

  A Symbol encodes two fields of information, a *name* and a *Content Dictionary*. Each is a sequence of characters matching a regular expression, as described below.

  A Symbol encodes three fields of information, a *name*, a *Content Dictionary*, and (optionally) a role. The name of a symbol is a sequence of characters matching the regular expression described in Section 3.3. The Content Dictionary is the location of the definition of the symbol, consisting of a name (a sequence of characters matching the regular expression described in Section 3.3) and, optionally, a unique prefix called a *uristub* which is used to disambiguate multiple Content Dictionaries of the same name. The role is a restriction on where the symbol may appear in an *OpenMath* object. The possible roles are described in Section 3.1.4.

- (vi) Variable.

  A Variable consists of must have a *name* which is a sequence of characters matching a regular expression, as described in Section 3.3. Where the variable is a member of an enumerated set it also has a child which is an *OpenMath* object representing its indices.

3.1.2 Derived *OpenMath* Objects

A derived *OpenMath* object is built as follows:

- (i) If A is *not* an *OpenMath* object, then **foreign**(A) is an *OpenMath foreign object*.

3.1.3 Compound *OpenMath* Objects

OpenMath objects are built recursively as follows.

- (i) Basic *OpenMath* objects are *OpenMath* objects. Derived *OpenMath* objects are *not* *OpenMath* objects.

- (ii) If $A_1, \ldots, A_n$ $(n > 0)$ are *OpenMath* objects, then
$$\textbf{application}\big(A_1, \ldots, A_n\big)$$
is an *OpenMath application object*.

- (iii) If $S_1, \ldots, S_n$ are *OpenMath* symbols, and ~~$A, A_1, \ldots, A_n, (n > 0)$ are *OpenMath* objects, then~~ A is an *OpenMath* object, and $A_1, \ldots, A_n, (n > 0)$ are *OpenMath* objects or *OpenMath* derived objects, then

$$\mathbf{attribution}\left(A, S_1\ A_1, \ldots, S_n\ A_n\right)$$

is an *OpenMath attribution object.* ~~and~~ A is the object *stripped of attributions.* $S_1, \ldots, S_n$ are referred to as *keys* and $A_1, \ldots, A_n$ as their associated *values*. The operation of recursively applying stripping to the stripped object is called *flattening of the attribution*. When the stripped object after flattening is a variable, the attributed object is called *attributed variable*.

- (iv) If B and C are *OpenMath* objects, and $v_1, \ldots, v_n\ (n \geq 0)$ are *OpenMath* variables or attributed variables, then

$$\mathbf{binding}\left(B, v_1, \ldots, v_n, C\right)$$

is an *OpenMath binding object*.

- (v) If S is an *OpenMath* symbol and $A_1, \ldots, A_n\ (n \geq 0)$ are *OpenMath* objects, then

$$\mathbf{error}\left(S, A_1, \ldots, A_n\right)$$

is an *OpenMath error object*.

3.1.4 *OpenMath* Symbol Rôles

The *rôle* of an *OpenMath* symbol is a restriction on where it can appear in an *OpenMath* object. A symbol cannot have more than one role. If no role is indicated then the symbol can be used anywhere. Possible roles are:

i. *binder* The symbol may only appear as the first child of an *OpenMath* binding object.

ii. *attribution* The symbol may only be used as key in an *OpenMath* attribution object, i.e. as the first element of a key-value pair, or in an equivalent context (for example to refer to the value of an attribution). This form of attribution may be ignored by an application, so should be used for information which does not change the meaning of the attributed *OpenMath* object.

iii. *semantic-attribution* This is the same as *attribution* except that it modifies the meaning of the attributed *OpenMath* object and thus cannot be ignored by an application.

iv. *error* The symbol can only appear as the first child of an *OpenMath* error object.

v. *default* The symbol can appear anywhere not defined in the previous four cases.

3.2 Further Description of *OpenMath* Objects

Informally, an *OpenMath object* can be viewed as a tree and is also referred to as a term. The objects at the leaves of *OpenMath* trees are called *basic objects*. The basic objects supported by *OpenMath* are:

Integer

    Arbitrary Precision integers.

Float

OpenMath floats are ieee 754 Double precision floating-point numbers. Other types of floating point number may be encoded in *OpenMath* by the use of suitable content dictionaries.

Character strings

are sequences of characters. These characters come from the Unicode standard [15].

Bytearrays

are sequences of bytes. There is no "byte" in *OpenMath* as an object of its own. However, a single byte can of course be represented by a bytearray of length 1. The difference between strings and bytearrays is the following: a character string is a sequence of bytes with a fixed interpretation (as characters, Unicode texts may require several bytes to code one character), whereas a bytearray is an uninterpreted sequence of bytes with no intrinsic meaning. Bytearrays could be used inside *OpenMath* errors to provide information to, for example, a debugger; they could also contain intermediate results of calculations, or `handles' into computations or databases.

Symbols

are uniquely defined by the Content Dictionary in which they occur and by a name. ~~In definition in Section 3.1 we have left this information implicit. However, it should be kept in mind that all symbols appearing in an *OpenMath* object are defined in a Content Dictionary.~~ The form of these definitions is explained in Chapter 5. Each symbol has no more than one definition in a Content Dictionary. Many Content Dictionaries may define differently a symbol with the same name (e.g. the symbol `union` is defined as associative -commutativeset theoretic union in a Content Dictionary `set1` but another Content Dictionary, `multiset1` might define a symbol `union` as the union of multi -sets). ~~The name of a symbol can only contain alphanumeric characters and underscores. More precisely, a symbol name matches the following regular expression:~~

~~[A-Za-z][A-Za-z0-9_]*~~

~~Notice that these symbol names are case sensitive. *OpenMath recommends* that symbol names should be no longer than 100 characters.~~

Variables

are meant to denote parameters, variables or indeterminates (such as bound variables of function definitions, variables in summations and integrals, independent variables of derivatives). ~~Plain variable names are restricted to use a subset of the printable ASCII characters. Formally the names must match the regular expression:~~

~~[A-Za-z0-9+(),-./:?!#$%*;=@[]^_`{|}]+~~

A variable may have one or more children which are themselves *OpenMath* objects and are treated as scripts. Thus it is possible to create a variable with cardinality such as x_1 or x_i.

Currently there is one way of making a derived *OpenMath* object.

Foreign

The four following constructs can be used to make compound *OpenMath* objects.

Application

constructs an *OpenMath* object from a sequence of one or more *OpenMath* objects. The first argument of application is referred to as "head" while the remaining objects are called "arguments". An *OpenMath* application object can be used to convey the mathematical notion of application of a function to a set of arguments. For instance, suppose that the *OpenMath* symbol sin is defined in a Content Dictionary for trigonometry, then **application**(sin, x) is the abstract *OpenMath* object corresponding to $\sin(x)$. More generally, an *OpenMath* application object can be used as a constructor to convey a mathematical object built from other objects such as a polynomial constructed from a set of monomials. Constructors build inhabitants of some symbolic type, for instance the type of rational numbers or the type of polynomials. The rational number, usually denoted as $1/2$, is represented by the *OpenMath* application object **application**(Rational, 1, 2). The symbol Rational must be defined, by a Content Dictionary, as a constructor symbol for the rational numbers.
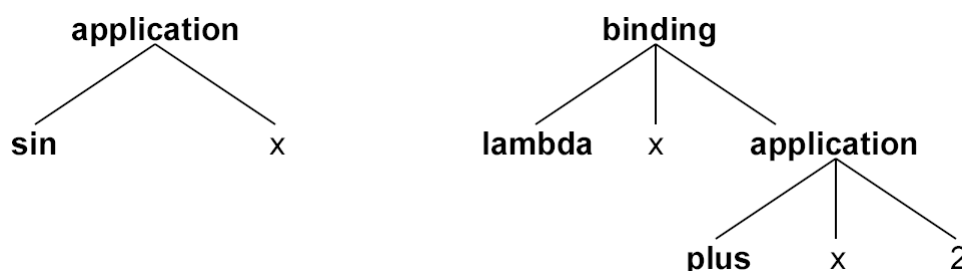
Figure 3.1 The *OpenMath* application and binding objects for $\sin(x)$ and $\lambda x.x + 2$ in tree-like notation.

Binding

objects are constructed from an *OpenMath* object, and from a sequence of zero or more variables followed by another *OpenMath* object. The first *OpenMath* object is the "binder" object. Arguments 2 to $n - 1$ are always variables to be bound in the "body" which is the n^{th} argument object. It is allowed to have no bound variables, but the binder object and the body should be present. Binding can be used to express functions or logical statements. The function $\lambda x.x + 2$, in which the variable x is bound by λ, corresponds to a binding object having as binder the *OpenMath* symbol lambda:

$$\mathbf{binding}(\text{lambda}, x, \mathbf{application}(\text{plus}, x, 2)).$$

Phrasebooks are allowed to use α conversion in order to avoid clashes of variable names. Suppose an object Ω contains an occurrence of the object **binding**(B, v, C). This object **binding**(B, v, C) can be replaced in Ω by **binding**(B, z, C') where z is a variable not occurring free in C and C' is obtained from C by replacing each free (i.e., not bound by any intermediate **binding** construct) occurrence of v by z. This operation preserves the semantics of the object Ω. In the above example, a phrasebook is thus allowed to transform the object to, e.g.

$$\mathbf{binding}(\text{lambda}, v, \mathbf{binding}(\text{lambda}, z, \mathbf{application}(\text{times}, z, z))).$$
$$\mathbf{binding}(\text{lambda}, z, \mathbf{application}(\text{plus}, z, 2)).$$

Repeated occurrences of the same variable in a binding operator are allowed. An *OpenMath* application should treat a binding with multiple occurrences of the same variable as equivalent to the binding in which all but the last occurrence of each variable is replaced by a new variable which does not occur free in the body of the binding.

$$\textbf{binding}(\text{lambda}, v, v, \textbf{application}(\text{times}, v, v))$$

is semantically equivalent to:

$$\textbf{binding}(\text{lambda}, v', v, \textbf{application}(\text{times}, v, v))$$

so that the resulting function is actually a constant in its first argument (v' does not occur free in the body **application**(times, v, v))).

Attribution

decorates an object with a sequence of one or more pairs made up of an *OpenMath* symbol, the "attribute", and an associated ~~*OpenMath*~~ object, the "value of the attribute" . The value of the attribute can be an *OpenMath* attribution object itself. As an example of this, consider the *OpenMath* objects representing groups, automorphism groups, and group dimensions. It is then possible to attribute an *OpenMath* object representing a group by its automorphism group, itself attributed by its dimension.

OpenMath objects can be attributed with *OpenMath* derived objects, which are containers for non-*OpenMath* structures. For example a mathematical expression could be attributed with its spoken or visual rendering.

Composition of attributions, as in

$$\textbf{attribution}\big(\textbf{attribution}(A, S_1\ A_1, \dots, S_h\ A_h), S_{h+1}\ A_{h+1}, \dots, S_n\ A_n\big)$$

is semantically equivalent to a single attribution, that is

$$\textbf{attribution}\big(A, S_1\ A_1, \dots, S_h\ A_h, S_{h+1}\ A_{h+1}, \dots, S_n\ A_n\big).$$

The operation that produces an object with a single layer of attribution is called *flattening*.

Multiple attributes with the same name are allowed. While the order of the given attributes does not imply any notion of priority, potentially it could be significant. For instance, consider the case in which $S_h = S_n$ $(h < n)$ in the example above. Then, the object is to be interpreted as if the value A_n overwrites the value A_h. (*OpenMath* however does not mandate that an application preserves the attributes or their order.)

Attribution acts as either adornment annotation or as semantical annotation. When the key has rôle *attribution*, then replacement of the attributed object by the object itself is not harmful and preserves the semantics. When the key has r ôle *semantic-attribution* then the attributed object is modified by the attribution and cannot be viewed as semantically equivalent to the stripped object. If the attribute lacks the r ôle specification then attribution is acting as adornment annotation.

Objects can be decorated in a multitude of ways. In [3], typing of *OpenMath* objects is expressed by using an attribution. The object **attribution**(A, type t) represents the judgment stating that object A has type t. Note that both A and t are *OpenMath* objects.

~~Attribution can act as either annotation, in the sense of adornment, or as modifier. In the former case, replacement of the adorned object by the object itself is probably not harmful (preserves the semantics). In the latter case however, it may very well be. Therefore, attribution in general should by default be treated as a construct rather than as adornment. Only when the CD definitions of the attributes make it clear that they are adornments, can the attributed object be viewed as semantically equivalent to the stripped object.~~

Error

>   is made up of an *OpenMath* symbol and a sequence of zero or more *OpenMath* objects. This
>   object has no direct mathematical meaning. Errors occur as the result of some treatment on an
>   *OpenMath* object and are thus of real interest only when some sort of communication is taking
>   place. Errors may occur inside other objects and also inside other errors. Error objects might
>   consist only of a symbol as in the object: **error**(S).

3.3 Names

The names of symbols, variables and content dictionaries must conform to the following rules, which
are designed to be compatible with standards such as Unicode and XML. These standards group
individual characters into letters, digits, combining characters and extenders. Informally, these are
defined as follows:

Letters

>   are elements of an alphabet such as latin, cyrillic, kanji etc.

Digits

>   are atomic numbers, from which compound numbers can be constructed (for example `1' is a
>   digit but `11' is not).

Combining Characters

>   are used to combine several characters to produce a new one, for example an accented
>   character.

Extenders

>   are characters which are neither letters nor combining characters but modify the appearance of
>   other characters in some way.

Formally, we use the precise definitions given in the Unicode standard [15].

Then a legal *OpenMath* name is defined by the following grammar:

>   Name → (Letter | '_') (Char)*
>   Char → Letter | Digit | '.' | '-' | '_' | CombiningChar | Extender

Uristubs Uristubs must conform to the grammar described in [9]. Note that if non-ASCII characters
are used in a CD or symbol name then when a URI for that symbol is constructed it will be necessary
to map the non -ASCII characters to a sequence of octets. The precise mechanism for doing this
depends on the URI scheme.

Note on content dictionary names It is a common convention to store a Content Dictionary in a file
of the same name, which can cause difficulties on many file systems. If this convention is to be
followed then *OpenMath recommends* that the name be restricted to the subset of the above grammar
which is a legal POSIX [8] filename, namely:

>   Name → (PosixLetter | '_') (Char)*

Char   → PosixLetter | Digit | '.' | '-' | '_'
PosixLetter → 'a' | 'b' | ... | 'z' | 'A' | 'B' | ... | 'Z'

3.4 Summary

- *OpenMath* supports basic objects like integers, symbols, floating-point numbers, character strings, bytearrays, and variables.

- *OpenMath* compound objects are of four kinds: applications, bindings, errors, and attributions.

- *OpenMath* objects may be attributed with non-*OpenMath* objects via the use of derived *OpenMath* objects.

- *OpenMath* objects have the expressive power to cover all areas of computational mathematics.

Observe that an *OpenMath* application object is viewed as a "tree" by software applications that do not understand Content Dictionaries, whereas a Phrasebook that understands the semantics of the symbols, as defined in the Content Dictionaries, should interpret the object as functional application, constructor, or binding accordingly. Thus, for example, for some applications, the *OpenMath* object corresponding to $2 + 5$ may result in a command that writes 7.

Chapter 4
OpenMath Encodings

In this chapter, two encodings are defined that map between *OpenMath* objects and byte streams. These byte streams constitute a low level representation that can be easily exchanged between processes (via almost any communication method) or stored and retrieved from files.

The first encoding uses ISO 646:1983 characters [10] (also known as ascii characters) and is an xml application. Although the xml markup of the encoding uses only ascii characters, *OpenMath* strings may use arbitrary Unicode/ISO 10646:1988 characters [15]. It can be used, for example, to send *OpenMath* objects via e-mail, news, cut-and-paste, etc. The texts produced by this encoding can be part of xml documents.

The first encoding is a character-based encoding in xml format. In previous versions of the *OpenMath* Standard this encoding was a restricted subset of the full legal xml syntax. In this version, however, we have removed all these restrictions so that the earlier encoding is a strict subset of the existing one. The xml encoding can be used, for example, to send *OpenMath* objects via e-mail, cut-and-paste, etc. and to embed *OpenMath* objects in xml documents or to have *OpenMath* objects processed by xml-aware applications.

The second encoding is a binary encoding that is meant to be used when the compactness of the encoding is important (interprocess communications over a network is an example).

Note that these two encodings are sufficiently different for autodetection to be effective: an application reading the bytes can very easily determine which encoding is used.

4.1 The xml Encoding

This encoding has been designed with two main goals in mind:

1. to provide an encoding that uses common character sets (so that it can be easily included in

most documents and transport protocols) and that is both readable and writable by a human.

2. to provide an encoding that can be included (embedded) in   xml documents or processed by xml-aware applications.

4.1.1 A ~~Grammar~~Schema for the xml Encoding

The xml encoding of an *OpenMath* object is defined by the Relax NG schema   [12] given below. Relax NG has a number of advantages over the older XSD Schema format   [16], in particular it allows for tighter control of attributes and has a modular, extensible structure. Although we have made the xml form of the schema normative it can be generated using the compact syntax given in Appendix B. It is also very easy to restrict the schema to allow a limited set of *OpenMath* symbols as described in Appendix C.

Standard tools exist for generating a DTD or an XSD schema from a Relax NG Schema. Examples of such documents are given in Appendix E and Appendix D respectively.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for OpenMath 2 -->
<grammar xmlns:xlink="http://www.w3.org/1999/xlink"
         ns="http://www.openmath.org/OpenMath"
         xmlns="http://relaxng.org/ns/structure/1.0"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <!-- OM2: allow OMR -->
  <define name="omel">
    <choice>
      <ref name="OMS"/>
      <ref name="OMV"/>
      <ref name="OMI"/>
      <ref name="OMB"/>
      <ref name="OMSTR"/>
      <ref name="OMF"/>
      <ref name="OMA"/>
      <ref name="OMBIND"/>
      <ref name="OME"/>
      <ref name="OMATTR"/>
      <ref name="OMR"/>
    </choice>
  </define>
  <!-- things which can be variables -->
  <define name="omvar">
    <choice>
      <ref name="OMV"/>
      <ref name="attvar"/>
    </choice>
  </define>
  <define name="attvar">
    <element name="OMATTR">
      <ref name="common.attributes"/>
      <group>
        <ref name="OMATP"/>
        <choice>
          <ref name="OMV"/>
          <ref name="attvar"/>
        </choice>
      </group>
    </element>
  </define>
  <!-- OM2: common.attributes attributes -->
  <define name="common.attributes">
    <optional>
      <attribute name="id">
        <data type="ID"/>
```

```xml
      </attribute>
    </optional>
</define>
<!-- symbol -->
<define name="OMS">
  <element name="OMS">
    <ref name="common.attributes"/>
    <ref name="attlist.OMS"/>
  </element>
</define>
<define name="attlist.OMS">
  <attribute name="name">
    <data type="NCName"/>
  </attribute>
  <attribute name="cd">
    <data type="NCName"/>
  </attribute>
  <attribute name="uristub">
    <data type="anyURI"/>
  </attribute>
</define>
<!-- variable -->
<define name="OMV">
  <element name="OMV">
    <ref name="common.attributes"/>
    <ref name="attlist.OMV"/>
  </element>
</define>
<define name="attlist.OMV">
  <attribute name="name">
    <data type="NCName"/>
  </attribute>
</define>
<!-- integer -->
<define name="OMI">
  <element name="OMI">
    <ref name="common.attributes"/>
    <data type="string">
      <param name="pattern">\s*(-\s?)?[0-9]+(\s[0-9]+)*\s*</param>
    </data>
  </element>
</define>
<!-- byte array -->
<define name="OMB">
  <element name="OMB">
    <ref name="common.attributes"/>
    <data type="base64Binary"/>
  </element>
</define>
<!-- string -->
<define name="OMSTR">
  <element name="OMSTR">
    <ref name="common.attributes"/>
    <text/>
  </element>
</define>
<!-- floating point -->
<define name="OMF">
  <element name="OMF">
    <ref name="common.attributes"/>
    <ref name="attlist.OMF"/>
  </element>
</define>
<define name="attlist.OMF">
  <choice>
    <attribute name="dec">
      <data type="string">
        <param name="pattern">(-?)([0-9]+)?(\.[0-9]+)?(e([+\-]?)[0-9]+)?</param>
      </data>
    </attribute>
    <attribute name="hex">
```

```
      <data type="string">
        <param name="pattern">[0-9A-F]+</param>
      </data>
    </attribute>
  </choice>
</define>
<!-- apply constructor -->
<define name="OMA">
  <element name="OMA">
    <ref name="common.attributes"/>
    <oneOrMore>
      <ref name="omel"/>
    </oneOrMore>
  </element>
</define>
<!-- binding constructor and variable -->
<define name="OMBIND">
  <element name="OMBIND">
    <ref name="common.attributes"/>
    <ref name="omel"/>
    <ref name="OMBVAR"/>
    <ref name="omel"/>
  </element>
</define>
<define name="OMBVAR">
  <element name="OMBVAR">
    <ref name="common.attributes"/>
    <oneOrMore>
      <ref name="omvar"/>
    </oneOrMore>
  </element>
</define>
<!-- error -->
<define name="OME">
  <element name="OME">
    <ref name="common.attributes"/>
    <ref name="OMS"/>
    <zeroOrMore>
      <ref name="omel"/>
    </zeroOrMore>
  </element>
</define>
<!-- attribution constructor and attribute pair constructor -->
<define name="OMATTR">
  <element name="OMATTR">
    <ref name="common.attributes"/>
    <ref name="OMATP"/>
    <ref name="omel"/>
  </element>
</define>
<!-- OM2: allow OMFOREIGN -->
<define name="OMATP">
  <element name="OMATP">
    <ref name="common.attributes"/>
    <oneOrMore>
      <ref name="OMS"/>
      <choice>
        <ref name="omel"/>
        <ref name="OMFOREIGN"/>
      </choice>
    </oneOrMore>
  </element>
</define>
<!-- OM2: OMFOREIGN -->
<define name="OMFOREIGN">
  <element name="OMFOREIGN">
    <ref name="common.attributes"/>
    <zeroOrMore>
      <choice>
        <ref name="omel"/>
        <ref name="notom"/>
```

```
        </choice>
      </zeroOrMore>
    </element>
  </define>
  <!-- Any elements not in the om namespace (valid om is allowed as a descendant) -->
  <define name="notom">
    <choice>
      <element>
        <anyName>
          <except>
            <nsName/>
          </except>
        </anyName>
        <zeroOrMore>
          <attribute>
            <anyName/>
          </attribute>
        </zeroOrMore>
        <zeroOrMore>
          <choice>
            <ref name="omel"/>
            <ref name="notom"/>
          </choice>
        </zeroOrMore>
      </element>
      <text/>
    </choice>
  </define>
  <!-- OM object constructor -->
  <define name="OMOBJ">
    <element name="OMOBJ">
      <ref name="common.attributes"/>
      <ref name="omel"/>
    </element>
  </define>
  <define name="attlist.OMOBJ">
    <attribute name="version">
      <choice>
        <value>1.0</value>
        <value>1.2</value>
        <value>2.0</value>
      </choice>
    </attribute>
  </define>
  <start>
    <ref name="OMOBJ"/>
  </start>
  <!-- OM2: OMR -->
  <define name="OMR">
    <element name="OMR">
      <ref name="common.attributes"/>
      <ref name="attlist.OMR"/>
    </element>
  </define>
  <define name="attlist.OMR">
    <attribute name="xlink:href"/>
    <attribute name="xlink:type">
      <value>simple</value>
    </attribute>
    <attribute name="xlink:show">
      <value>embed</value>
    </attribute>
  </define>
</grammar>
```

~~the xml encoding of an *OpenMath* object is defined by the dtd given in Figure 4.1 below with the following additional rules not implied by the xml dtd.~~

- Comments are permitted only between elements, not within element character data.

- Processing Instructions are only allowed before the OMOBJ element.

- The content of an OMB element, is a valid base64-encoded text.

- The character data forming element content and attribute-values matches the regular expressions of .

DTD for the *OpenMath* xml encoding of objects.

```
<!-- DTD for OM Objects - sb 29.10.98 -->
<!-- sb 3.2.99 -->

<!--
    general list of embeddable elements
      · excludes OMATP as this is only embeddable in OMATTR
      · excludes OMBVAR as this is only embeddable in OMBIND
-->

<!ENTITY % omel "OMS | OMV | OMI | OMB | OMSTR
                              | OMF | OMA | OMBIND | OME
                              | OMATTR | ">

<!-- things which can be variables -->

<!ENTITY % omvar "OMV | OMATTR" >



<!-- symbol -->
<!ELEMENT OMS EMPTY>

  <!ATTLIST OMS
               name CDATA #REQUIRED
               cd CDATA #REQUIRED >

<!-- variable -->
<!ELEMENT OMV EMPTY>
<!ATTLIST OMV
               name CDATA #REQUIRED >

<!-- integer -->
<!ELEMENT OMI (#PCDATA) >


<!-- byte array -->
<!ELEMENT OMB (#PCDATA) >


<!-- string -->
<!ELEMENT OMSTR (#PCDATA) >


<!-- floating point -->
<!ELEMENT OMF EMPTY>
<!ATTLIST OMF
               dec CDATA #IMPLIED
               hex CDATA #IMPLIED>

<!-- apply constructor -->
<!ELEMENT OMA (%omel;)+ >



<!-- binding constructor & bound variables -->
```

```
<!ELEMENT OMBIND ((%omel;), OMBVAR, (%omel;)) >


<!ELEMENT OMBVAR (%omvar;)+ >


<!-- error -->
<!ELEMENT OME (OMS, (%omel;)* ) >


<!-- attribution constructor & attribute pair constructor -->
<!ELEMENT OMATTR (OMATP, (%omel;)) >


<!ELEMENT OMATP (OMS, (%omel;))+ >




<!-- OM object constructor -->
<!ELEMENT OMOBJ (%omel;) >
<!ATTLIST OMOBJ
                xmlns:xlink CDATA #FIXED 'http://www.w3.org/1999/xlink'>
```

In addition, if the xml document encoding the *OpenMath* object is linearised into the xml concrete syntax, the following further constraints apply, which ensure thet the encoding may be read by *OpenMath* applications that may not include a full xml parser.

- The document should use utf-8 encoding.

- A <!DOCTYPE declaration should not be used.

- Character references should not be used. As <!DOCTYPE is not used, the only entity references that are allowed are the five predefined entity references: ' ('), " ("), < (<), > (>), & (&).

- The xml empty element form <|.../> should always be used to encode elements such as omf which are specified in the dtd as being empty. It should never be used for elements that may sometimes be empty, such as omstr.

Such a linearisation of an xml encoded *OpenMath* Object would match the match the character based grammar given in .

The notation used in this section and in _ should be quite straightforward (+ meaning "one or more", ? meaning zero or one, and | meaning "or"). The start symbol of the grammar is "start", "space" stands for the space character, "cr" for the carriage return character, "nl" for the line feed character and "tab" for the horizontal tabulation character.

Grammar for the xml encoding of *OpenMath* objects.

S   → (space|tab|cr|nl)+

integer → (−S?)? [0-9]+(S [0-9]+)*|(−S?)? x S? [0-9A-F]+(S [0-9A-F]+)*

edname   → [a-z][a-z0-9_]*

symbname → [A-Za-z][A-Za-z0-9_]*

fpdec   → (−?)([0-9]+)?(.[0-9]+)?(e([+-]?)[0-9]+)?

fphex   → [0-9ABCDEF]+

varname   → ([A-Za-z0-9+-=(), ./:?!#$%*;@[]^_`{|}])+

base64   → ([A-Za-z0-9 +/=]|S)+

char   → xml Character Data

symbnameatt → name S? = S? (" symbname " | ' symbname ')

ednameatt   → cd S? = S? (" edname " | ' edname ')

varnameatt → name S? = S? (" varname " | ' varname ')

fpdecatt   → dec S? = S? (" fpdec " | ' fpdec ')

fphexatt   → hex S? = S? (" fphex " | ' fphex ')

PI   → <? char ?>

comment   → <!-- char -->

SC   → S+ | (comment S)+

start   → (SC | PI)* <OMOBJ S?> S? object S? </OMOBJ S?>

symbol   → <OMS [[(S symbnameatt) (S ednameatt)]] S? />

variable   → <OMV varnameatt S? />
            | <OMATTRx S?> SC? omatp SC? variable SC? </OMATTR S?>

omatp   → <OMATP S?> SC? attrs SC? </ #1 S?>

object   → symbol
            | variable
            | <OMI S > S? integer S? </OMI S?>
            | <OMF S fpdecatt S?/ >
            | <OMF S fphexatt S?/ >
            | <OMSTR S?> char </OMSTR S?>
            | <OMB S?> base64 </OMB S?>
            | <OMA S?> SC? object SC? objects SC? </OMA S?>
            | <OMBIND S?> SC? object SC?
               <OMBVAR S?> SC? variables SC? </OMBVAR S?>
               SC? object SC? </OMBIND S?>
            | <OME S?> SC? symbol SC? objects SC? </OME S?>
            | <OMATTR S?> SC? <OMATP S?> SC? attrs SC? </OMBVAR S?>
               SC? object SC? </OMATTR S?>

attrs   → symbol S? object
            | symbol S? object S? attrs

objects   → SC?
            | object SC? objects

variables   → SC?
            | variable SC? variables

4.1.2 Informal description of the ~~Grammar~~ xml Encoding

An encoded *OpenMath* object is placed inside an `OMOBJ` element. This element can contain the elements (and integers) described above. It can take an optional `version` xml-attribute which indicates to which version of the *OpenMath* standard it conforms. This would allow an older *OpenMath* application which, for example, could not parse the full xml syntax, to accept objects with version "1" or "1.1" but reject objects with version "2".

We briefly discuss the xml encoding for each type of *OpenMath* object starting from the basic objects.

Integers

are encoded using the `OMI` element around the sequence of their digits in base 10 or 16 (most significant digit first). White space may be inserted between the characters of the integer representation, this will be ignored. After ignoring white space, integers written in base 10 match the regular expression `-?[0-9]+`. Integers written in base 16 match `-?x[0-9A-F]+`. The integer 10 can be thus encoded as `<OMI> 10 </OMI>` or as `<OMI> xA </OMI>` but neither `<OMI> +10 </OMI>` nor `<OMI> +xA </OMI>` can be used.

The negative integer -120 can be encoded as either as decimal `<OMI> -120 </OMI>` or as hexadecimal `<OMI> -x78 </OMI>`.

Symbols

are encoded using the `OMS` element. This element has ~~two~~ three xml-attributes `cd`, `name`, and `uristub`. The value of `cd` is the name of the Content Dictionary in which the symbol is defined and the value of `name` is the name of the symbol. The optional `uristub` attribute is a URI that can be used to disambiguate between two content dictionaries with the same name. ~~The name of the Content Dictionary is compulsory, but a future revision of the OpenMath standard might introduce a defaulting mechanism.w~~ For example:

```
<OMS uristub="http://www.openmath.org/cd"
cd="transc" name="sin"/>
```

is the encoding of the symbol named `sin` in the Content Dictionary named `transc`, which is part of the collection maintained by the *OpenMath* Society.

The three attributes of the `OMS` can be used to build a URI for the symbol, for use in contexts where URI -based referencing mechanisms are used. This canonical URI is constructed as follows:

```
URI = uristub-value + '/' + cd-value + '/' +
name-value
```

For example

```
<OMS name="plus" cd="arith1" uristub="http://www.openmath.org/cd"/>
```

gives the URI `"http://www.openmath.org/cd/arith1/plus"` This
would allow us to refer uniquely to an openmath symbol from a MathML document [19].

```
<mathml:csymbol xmlns:mathml="http://www.w3.org/TR/MathML2/"
                definitionURL="http://www.openmath.org/CDs/arith1/plus">Z</csy
```

Note that the role attribute described in Section 3.1.4 is contained in the Content Dictionary
and is not part of the encoding of a symbol.

Variables

are encoded using the OMV element, with only one xml-attribute, name, whose value is the
variable name. ~~The variable name is a subset of the printable   ascii set of characters. In   particular, neither spaces nor double -quote " are allowed in variable names.~~ For instance, the
encoding of the object representing the variable x is: `<OMV name="x"/>`

For an enumerated variable the index or indices of the variable are encoded as a child. So for
example the encoding of the object representing the variable x_{i+1} is:

```
<OMV name="x">
  <OMA>
    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
    <OMV name="i"/>
    <OMI>1</OMI>
  </OMA>
</OMV>
```

Floating-point numbers

are encoded using the OMF element that has either the xml-attribute dec or the xml-attribute
hex. The two   xml-attributes cannot be present simultaneously. The value of   dec is the
floating-point number expressed in base 10, using the common syntax:

$$(-?)([0-9]+)?(\text{"."}[0-9]+)?(e(-?)[0-9]+)?.$$

The value of hex is the digits of the floating -point number expressed in base 16, with digits
0-9, A-F (mantissa, exponent, and sign from lowest to highest bits) using a least significant
byte ordering. For example,   `<OMF dec="1.0e -10"/>` is a valid floating   -point
number.

Character strings

are encoded using the OMSTR element. Its content is a Unicode text ~~(The default encoding is~~

characters `<` and `&` need to be represented by the entity references `<` and `&` respectively.

Bytearrays

are encoded using the `OMB` element. Its content is a sequence of characters that is a base64 encoding of the data. The base64 encoding is defined in rfc 1521 [2]. Basically, it represents an arbitrary sequence of octets using 64 "digits" (`A` through `Z`, `a` through `z`, `0` through `9`, `+` and `/`, in order of increasing value). Three octets are represented as four digits (the `=` character for padding to the right at the end of the data). All line breaks and carriage return, space, form feed and horizontal tabulation characters are ignored. The reader is refered to [2] for more detailed information.

In detail the encoding of an *OpenMath* object is described below.

Applications

are encoded using the `OMA` element. The application whose root is the *OpenMath* object e_0 and whose arguments are the *OpenMath* objects e_1, …, e_n is encoded as `<OMA>` C_0 C_1… C_n `</OMA>` where C_i is the encoding of e_i.

For example, **application**(sin, x) is encoded as:

```
<OMA>
  <OMS uristub="http://www.openmath.org/cd" cd="transc1" name="sin"/>
  <OMV name="x"/>
</OMA>
```

provided that the symbol `sin` is defined to be a function symbol in a Content Dictionary named `transc1`.

Binding

is encoded using the `OMBIND` element. The binding by the *OpenMath* object b of the *OpenMath* variables x_1, x_2, …, x_n in the object c is encoded as `<OMBIND>` B `<OMBVAR>` X_1 … X_n `</OMBVAR>` C `</OMBIND>` where B, C, and X_i are the encodings of b, c and x_i, respectively.

For instance the encoding of **binding**(lambda, x, **application**(sin, x)) is:

```
<OMBIND>
  <OMS uristub="http://www.openmath.org/cd" cd="fns1" name="lambda"/>
  <OMBVAR>
    <OMV name="x"/>
  </OMBVAR>
  <OMA>
    <OMS uristub="http://www.openmath.org/cd" cd="transc1" name="sin"/>
```

```
                <OMV name="x"/>
            </OMA>
        </OMBIND>
```

Binders are defined in Content Dictionaries, in particular, the symbol `lambda` is defined in the Content Dictionary `fns1` for functions over functions.

Attributions

are encoded using the OMATTR element. If the *OpenMath* object e is attributed with (s_1, e_1), …, (s_n, e_n) pairs (where s_i are the attributes), it is encoded as `<OMATTR> <OMATP>` S_1 C_1 … S_n C_n `</OMATP>` E `</OMATTR>` where S_i is the encoding of the symbol s_i, C_i of the object e_i and E is the encoding of e.

Examples are the use of attribution to decorate a group by its automorphism group:

```
        <OMATTR>
            <OMATP>
                <OMS uristub="http://www.openmath.org/cd" cd="groups" name="autom
                [..group-encoding..]
            </OMATP>
            [..group-encoding..]
        </OMATTR>
```

or to express the type of a variable:

```
        <OMATTR>
            <OMATP>
                <OMS uristub="http://www.openmath.org/cd" cd="ecc" name="type" />
                <OMS uristub="http://www.openmath.org/cd" cd="ecc" name="real" />
            </OMATP>
            <OMV name="x" />
         </OMATTR>
```

A special use of attributions is to associate non-*OpenMath* data with an *OpenMath* object. This is done using the OMFOREIGN element. The children of this element must be well-formed xml. For example the attribution of the *OpenMath* object $\sin(x)$ with its representation in Presentation MathML is:

```
      <OMATTR>
        <OMATP>
          <OMS cd="presentation1" name="mathml"/>
          <OMFOREIGN>
            <math xmlns="http://www.w3.org/TR/MathML2">
              <mi>sin</mi><mfenced><mi>x</mi></mfenced>
            </math>
          </OMFOREIGN>
        </OMATP>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="transc1" name="sin"/>
          <OMV name="x"/>
```

```
        </OMA>
      </OMATTR>
```

Of course not everything has a natural XML encoding in this way and often the contents of a `OMFOREIGN` will just be data or some kind of encoded string. For example the attribution of the previous object with its LaTeX representation could be achieved as follows:

```
<OMATTR>
  <OMATP>
    <OMS cd="presentation1" name="latex"/>
    <OMFOREIGN>
      sin\,(x)
    </OMFOREIGN>
  </OMATP>
  <OMA>
    <OMS uristub="http://www.openmath.org/cd" cd="transc1" name="sin"/>
    <OMV name="x"/>
  </OMA>
</OMATTR>
```

Errors

are encoded using the `OME` element. The error whose symbol is s and whose arguments are the *OpenMath* objects $e_1, \ldots, e_n$ is encoded as `<OME>` C_s $C_1 \ldots C_n$ `</OME>` where C_s is the encoding of s and C_i the encoding of e_i.

If an `aritherror` Content Dictionary contained a `DivisionByZero` symbol, then the object **error**(DivisionByZero, **application**(divide, x, 0)) would be encoded as follows:

```
<OME>
<OMS uristub="http://www.openmath.org/cd" cd="aritherror" name="Divisio
<OMA>
    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="divide
    <OMV name="x"/>
    <OMI> 0 </OMI>
</OMA>
</OME>
```

References

OpenMath integers, floating point numbers, character strings, byearrays, applications, binding, attributions can also be encoded as an empty `OMR` element with an `xlink:href` attribute whose value is the value of an id attribute of an *OpenMath* object of that type. The *OpenMath* element represented by this `OMR` element is a copy of the *OpenMath* element pointed to in the `xlink:xref` attribute. Note that the representation of the `OMR` element is *structurally equal*, but not identical to the element it points to.

For instance, the *OpenMath* object
application(f, **application**(f, **application**(f, a, a), **application**(f, a, a)), **application**(f, **applica**

can be encoded in the xml encoding as either one of the xml encodings below (and some intermedidate versions as well).

```
    <OMOBJ>                                    <OMOBJ>
     <OMA>                                      <OMA>
       <OMA>                                      <OMA id="t1">
         <OMV name="f"/>                            <OMV name="f"/>
         <OMA>                                      <OMA id="t11">
           <OMV name="f"/>                            <OMV name="f"/>
           <OMV name="a"/>                            <OMV name="a"/>
           <OMV name="a"/>                            <OMV name="a"/>
         </OMA>                                     </OMA>
         <OMA>                                      <OMR xlink:href="t11"/>
           <OMV name="f"/>
           <OMV name="a"/>
           <OMV name="a"/>
         </OMA>                                    </OMA>
         <OMA>                                      <OMR xlink:href="t1"/>
           <OMV name="f"/>
           <OMA>
             <OMV name="f"/>
             <OMV name="a"/>
             <OMV name="a"/>
           </OMA>
           <OMA>
             <OMV name="f"/>
             <OMV name="a"/>
             <OMV name="a"/>
           </OMA>
         </OMA>
       </OMA>
     </OMA>
    </OMOBJ>                                    </OMOBJ>
```

Figure 4.1 Shared vs. unshared representations

We say that an *OpenMath* element dominates all its children and all elements they dominate. An OMR element dominates its target, i.e. the element that carries the `id` attribute pointed to by the `xref` attribute. For instance in the representation in Figure Figure 4.1, the OMA element with `id="t1"` and also the second OMR dominate the OMA element with `id="t11"`.

4.1.2.1 An Acyclicity Constraint

The occurrences of the OMR element must obey the following global *acyclicity constraint*: An *OpenMath* element may not dominate itself.

Consider for instance the following (illegal) xml representation

```
<OMOBJ>
  <OMA id="foo">
    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="divide"/>
    <OMI>1</OMI>
    <OMA>
      <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
      <OMI>1</OMI>
      <OMR xref="foo"/>
    </OMA>
  </OMA>
```

```
    </OMOBJ>
```

Here, the `OMA` element with `id="foo"` dominates its second child, which dominates the `OMR` element, which dominates its target: the element with `id="foo"`. So by transitivity, this element dominates itself, and by the acyclicity constraint, it is not the xml representation of an *OpenMath* element. Even though it could be given the interpretation of the continued fraction

$$\cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{...}}}$$

this would correspond to an infinite tree of applications, which is not admitted by the structure of *OpenMath* objects described in section 3.

Note that the acyclicity constraints is not restricted to such simple cases, as the following example shows.

```
<OMOBJ>                               <OMOBJ>
  <OMA id="bar">                        <OMA id="baz">
    <OMS cd="arith1" name="plus"/>        <OMS cd="arith1" name="plus"/>
    <OMI>1</OMI>                          <OMI>1</OMI>
    <OMR xref="baz"/>                     <OMR xref="bar"/>
  </OMA>                                </OMA>
</OMOBJ>                               </OMOBJ>
```

Here, the `OMA` with `id="bar"` dominates its second child, the `OMR` with `xref="baz"`, which dominates its target `OMA` with `id="baz"`, which in turn dominates its second child, the `OMR` with `xref="bar"`, this finally dominates its target, the original `OMA` element with `id="bar"`. So this pair of *OpenMath* objects violates the acyclicity constraint and is not the xml representation of an *OpenMath* object.

4.1.2.2 Sharing and Bound Variables

Note that the `OMR` element is a *syntactic* referencing mechanism: an `OMR` element stands for the exact xml element it points to. In particular, referencing does not interact with binding in a semantically intuitive way, since it allows for variable capture. Consider for instance the following xml representation:

```
<OMBIND id="outer">
  <OMS uristub="http://www.openmath.org/cd" cd="fns1" name="lambda"/>
  <OMBVAR><OMV name="X"/></OMBVAR>
  <OMA>
    <OMV name="f"/>
    <OMBIND id="inner">
      <OMS uristub="http://www.openmath.org/cd" cd="fns1" name="lambda"/>
      <OMBVAR><OMV name="X"/></OMBVAR>
      <OMR id="copy" xlink:href="orig"/>
    </OMBIND>
    <OMA id="orig"><OMV name="g"/><OMV name="X"/></OMA>
  </OMA>
</OMBIND>
```

it represents the *OpenMath* object

$$\textbf{binding}(\lambda, X, \textbf{application}(f, \textbf{binding}(\lambda, X, \textbf{application}(g, X)), \textbf{application}(g, X))))$$

which has two subterms of the form **application**(g, X), one with `id="orig"` (the one explicitly represented) and one with `id="copy"`, represented by the OMR element. In the original, the variable X is bound by the *outer* OMBIND element, and in the copy, the variable X is bound by the *inner* OMBIND element. We say that the inner OMBIND has captured the variable X.

It is well -known that variable capture does not conserve semantics. For instance, we could use α-conversion to rename the inner occurrence of x into - say - y arriving at the (same) object

$$\textbf{binding}(\lambda, X, \textbf{application}(f, \textbf{binding}(\lambda, Y, \textbf{application}(g, Y)), \textbf{application}(g, X))))$$

Using references that capture variables in this way can easily lead to representation errors, and is not recommended.

4.1.3 Embedding *OpenMath* in xml Documents

The above encoding of xml encoded *OpenMath* specifies the grammar to be used in files that encode a single *OpenMath* object, and specifies the character streams that a conforming *OpenMath* application should be able to accept or produce.

When embedding xml encoded *OpenMath* objects into a larger xml document one may wish, or need, to use other xml features. For example use of extra xml attributes to specify xml Namespaces [17] or xml:lang attributes to specify the language used in strings [18]. ~~Also, the encoding used in the larger document may not be utf-8.~~

In particular, if *OpenMath* is used with applications that use the xml Namespace Recommendation [17] then they should ensure that *OpenMath* elements are in the namespace `http://www.openmath.org/OpenMath`. This is most conveniently achieved by adding the namespace declaration

```
xmlns="http://www.openmath.org/OpenMath"
```

as an attribute to each OMOBJ element in the document.

Furthermore, for any *OpenMath* object that contains the OMR element, we have to add the XLink namespace declaration
`xmlns:xlink="http://www.w3.org/1999/xlink"`.

If such xml features are used then the xml application controlling the document must, if passing the *OpenMath* fragment to an *OpenMath* application, remove any such extra attributes and must ensure that the fragment is encoded according to the grammar specified above.

4.2 The Binary Encoding

The binary encoding was essentially designed to be more compact than the xml encodings, so that it can be more efficient if large amounts of data are involved. For the current encoding, we tried to keep the right balance between compactness, speed of encoding and decoding and simplicity (to allow a simple specification and easy implementations).

4.2.1 A Grammar for the Binary Encoding

id:*l*

start	→ [24] object [25]	| [24+64] object [25]
object	→ integer	
	| float	
	| variable	
	| indexed_variable	
	| symbol	
	| string	
	| bytearray	
	| derived	
	| construct	
	| internal_reference	
	| external_reference	
integer	→ [1] [_]	| [1+64] [*n*] id:*n* [_]
	| [1+128] {_}	| [1+64+128] {*n*} id:*n* {_}
	| [2] [*n*] [_] digits:*n*	| [2+64] [*n*] [*m*] [_] digits:*n* id:*m*
	| [2+128] {*n*} [_] digits:*n*	| [2+64+128] {*n*} {*n*} [_] digits:*n* id:*n*
float	→ [3] {_}{_}	| [3+64] [*n*] id:*n* {_}{_}
variable	→ [5] [*n*] varname:*n*	| [5+64] [*n*] [*m*] varname:*n* id:*m*
	| [5+128] {*n*} varname:*n*	| [5+64+128] {*n*} {*m*} varname:*n* id:*m*
indexed_variable	→ [10] [*n*] varname:*n* object [11]	[10+64] [*n*] [*m*] varname:*n* id:*m* object [11]
	[10+128] {*n*} varname:*n* object [11]	[10+64+128] {*n*} {*m*} varname:*n* id:*m* object [11]
symbol	→ [8] [*n*] [*m*] cdname:*n* symbname:*m*	| [8+64] [*n*] [*m*] [*k*] cdname:*n* symbname:*m* id:*k*
	[8+128] {*n*} {*m*} cdname:*n* symbname:*m*	[8+64+128] {*n*} {*m*} {*k*} cdname:*n* symbname:*m* id:*k*
	[9] [*n*] [*m*] [*k*] cdname:*n* symbname:*m* uri:*k*	[9+64] [*n*] [*m*] [*k*] [*l*] cdname:*n* symbname:*m* uri:*k* id:*l*
	[9+128] {*n*} {*m*} {*k*} cdname:*n* symbname:*m* uri:*k*	[9+64+128] {*n*} {*m*} {*k*} {*l*} cdname:*n* symbname:*m* uri:*k*
string	→ [6] [*n*] ~~chars~~ bytes:*n*	| [6+64] [*n*] ~~chars~~ bytes:*n*
	| [6+128] {*n*} ~~chars~~ bytes:*n*	[6+64+128] {*n*} {*m*} ~~chars~~ bytes:*n* id:*m*
	| [7] [*n*] ~~chars~~ bytes:2*n*	| [7+64] [*n*] [*m*] ~~chars~~ bytes:*n* id:*m*
	| [7+128] {*n*} ~~chars~~ bytes:2*n*	[7+64+128] {*n*} {*m*} ~~chars~~ bytes:2*n* id:*m*
bytearray	→ [4] [*n*] bytes:*n*	| [4+64] [*n*] [*m*] bytes:*n* id:*m*
	| [4+128] {*n*} bytes:*n*	| [4+64+128] {*n*} {*m*} bytes:*n* id:*m*
derived	→ [12] [*n*] bytes:*n*	| [12+64] [*n*] [*m*] bytes:*n* id:*m*
	| [12+128] {*n*} bytes:*n*	| [12+64+128] {*n*} {*m*} bytes:*n* id:*m*
construct	→ [16] object objects [17]	| [16+64] {*m*} id:*m* object objects [17]
	| [22] symbol objects [23]	| [22+64] {*m*} id:*m* symbol objects [23]

	[18] attrpairs object [19]	[18+64] {*m*} id:*m* attrpairs object [19]
	[26] object bvars object [27]	[26+64] {*m*} id:*m* object bvars object [27]
attrpairs	→ [20] pairs [21]	[20+64] {*m*} id:*m* pairs [21]
pairs	→ symbol object	
	symbol object pairs	
bvars	→ [28] vars [29]	[28+64] {*m*} id:*m* vars [29]
vars	→ attrvar	
	attrvar vars	
attrvar	→ variable	
	[18] attrpairs attrvar [19]	[18+64] {*m*} id:*m* attrpairs attrvar [19]
objects	→ object objects	
internal_reference	→ [30] [_]	
	[30+128] {_}	
external_reference	→ [31] [*n*] uri:*n*	
	[31+128] {*n*} uri:*n*	

Figure 4.2 Grammar of the binary encoding of *OpenMath* objects.

Figure Figure 4.2 gives a grammar for the binary encoding ("start" is the start symbol)..

The following conventions are used in this section: [*n*] denotes a byte whose value is the integer *n* (*n* can range from 0 to 255), {*m*} denotes four bytes representing the (unsigned) integer *m* in network byte order, [_] denotes an arbitrary byte, {_} denotes an arbitrary sequence of four bytes. ~~*name*:*n* denotes a sequence of *n* bytes named *name*. *name*:2*n* denotes a sequence of 2*n* bytes. "start" is the start symbol of the grammar.~~

xxxx:*n*, where *xxxx* is one of *symbname*, *cdname*, *varname*, *uri*, *id*, *digits*, or *bytes* denotes a sequence of *n* bytes that conforms to the constraints on *xxxx* strings. For instance, for *symbname*, *varname*, or *cdname*this is the regular expression described in   Section 3.3, for *uri* it is the grammar for URIs in [9].

4.2.2 Description of the Grammar

An *OpenMath* object is encoded as a sequence of bytes starting with the begin object tag (~~value 24~~ values 24 and 88) and ending with the end object tag (~~value 25~~ values 25 and 89). These are similar to the <OMOBJ> and </OMOBJ> tags of the xml encoding.

The encoding of each kind of   *OpenMath* object begins with a tag that is a single byte, holding a *token identifier* that describes the kind of object and two flags, the *long* flag and the *shared* flag. The identifier is stored in the first 6 bits (1 to 6). The long flag is the eighth bit   and the shared flag is the seventh bit. If the long flag is set, this signifies that the names, strings, and data fields in the encoded *OpenMath* object are longer than 255 byte or characters. The sharing flag indicates that the encoded object is shared in another (part of an) object somewhere else (see   Section 4.2.3). Note that if the sharing flag is set (in the right column of the grammar in Figure   Figure 4.2, then the encoding includes a representation of the identifier.

Here is a description of the binary encodings of every kind of *OpenMath* object:

Integers

are encoded depending on how large they are. There are four possible formats. Integers between -128 and 127 are encoded as the small integer (token identifier 1) followed by a single byte that is the value of the integer (interpreted as a signed character). For example 16 is encoded as $0x01 \ 0x10$. Integers between -2^{31} (-2147483648) and $2^{31} - 1$ (2147483647) are encoded as the small integer tag with the long flag set followed by the integer encoded in little endian format on four bytes (network byte order: the most significant byte comes first). For example, 128 is encoded as $0x81 \ 0x00000080$. The most general encoding begins with the big integer tag (token identifier 2) with the long flag set if the number of bytes in the encoding of the digits is greater or equal than 256. It is followed by the length (in bytes) of the sequence of digits, encoded on one byte (0 to 255, if the long flag was not set) or four bytes (network byte order, if the long flag was set). It is then followed by a byte describing the sign and the base. This 'sign/base' byte is $+$ (0x2B) or $-$ (0x2D) for the sign ored with the base mask bits that can be 0 for base 10 or 0x40 for base 16. It is followed by the strings of digits (as characters) in their natural order (as in the xml encoding). For example, 8589934592 (2^{33}) is encoded $0x02 \ 0x0A \ 0x2B$ $0x3835383939333435393932$ and xf

Symbols

are encoded as the symbol tags (token identifier 8) with the long flag set if the maximum of the length of the Content Dictionary name, ~~and~~ the symbol name or the URI stub is greater than or equal to 256 ~~(note that this should never be the case if the rules on symbols and Content Dictionary names are applied), followed by the length of the Content Dictionary name as a byte (if the long flag was not set) or a four byte integer (in network byte order) followed by the length of the symbol name as a byte (if the long flag was not set) or a four byte integer (in network byte order),, followed by the characters of the Content Dictionary name, followed by the characters of the symbol name.~~ . The symbol tags [8] and [8+128] are deprecated in *OpenMath*2 since symbols now have an optional role field, but are kept for backwards compatibility. The symbol tag is followed by the length of the Content Dictionary name, the symbol name, and the URI stub as a bytes (if the long flag was not set) or a four byte integers (in network byte order). These are followed by the characters of the Content Dictionary name, the symbol name, and the URI stub.

Variables

are encoded using the variable tags (token identifiers 5) with the long flag set if the number of bytes (characters) in the variable name is greater than or equal to 256 (this should never happen if the rules on variables are followed). Then, there is the number of characters as a byte (if the long flag was not set) or a four byte integer (in network byte order), followed by the characters of the name of the variable. For example, the variable x is encoded as $0x05$ $0x01 \ 0x78$.

Indexed Variables

are encoded using the indexed variable tags (token identifiers 10 and 11) as begin and end tags. The long flag set on the begin tag if the number of bytes (characters) in the variable name is greater than or equal to 256 (this should never happen if the rules on variables are followed). The start tag is followed by the number of characters as a byte (if the long flag was not set) or a four byte integer (in network byte order), followed by the characters of the name of the variable. This is followed by the encoding of an *OpenMath* object and the end tag [11].

Floating-point number

are encoded using the floating -point number tags (token identifier 3) followed by eight bytes that are the IEEE 754 representation [7], most significant bytes first. For example, 0.1 is encoded as `0x03 0x000000000000f03f`.

Character string

are encoded in two ways depending on whether the string contains utf-16 characters or not. If the string contains only 8 bit characters, it is encoded as the one byte character string tags (token identifier 6) with the long flag set if the number of bytes (characters) in the string is greater than or equal to 256. Then, there is the number of characters as a byte (if the length flag was not set) or a four byte integer (in network byte order), followed by the characters in the string. If the string contains two byte characters, it is encoded as the two byte character string tags (token identifier 7) with the long flag set if the number of characters in the string is greater or equal to 256. Then, there is the number of characters as a byte (if the long flag was not set) or a four byte integer (in network byte order), followed by the characters (utf-16 encoded Unicode).

Bytearrays

are encoded using the bytearray tags (token identifier 4) with the long flag set if the number ~~of bytes in the number of~~ elements is greater than or equal to 256. Then, there is the number of elements, as a byte (if the long flag was not set) or a four byte integer (in network byte order), followed by the elements of the arrays in their normal order.

Derived Objects

are encoded using the derived object tags (token identifier 12) with the long flag set if the number of bytes is greater than or equal to 256. Then, there is the number of elements, as a byte (if the long flag was not set) or a four byte integer (in network byte order), followed by the elements of the arrays in their normal order.

Applications

are encoded using the application tags (token identifier 16). More precisely, the application of E_0 to $E_1 \ldots E_n$ is encoded using the application tags (token identifier 16), the sequence of the encodings of E_0 to E_n and the end application tags (token identifier 17).

Bindings

are encoded using the binding tags (token identifier 26). More precisely, the binding by B of variables $V_1 \ldots V_n$ in C is encoded as the binding tags (token identifier 26), followed by the encoding of B, followed by the binding variables tags (token identifier 28), followed by the encodings of the variables $V_1 \ldots V_n$, followed by the end binding variables tags (token identifier 29), followed by the encoding of C, followed by the end binding tags (token identifier 27).

Attribution

are encoded using the attribution tags (token identifier 18). More precisely, attribution of the object E with (S_1, E_1), … (S_n, E_n) pairs (where S_i are the attributes) is encoded as the attributed object tags (token identifier 18), followed by the encoding of the attribute pairs as the attribute pairs tags (token identifier 20), followed by the encoding of each symbol and value, followed by the end attribute pairs tags (token identifier 21), followed by the encoding

of E, followed by the end attributed object tags (token identifier 19).

Error

are encoded using the error tags (token identifier 22). More precisely, S_0 applied to $E_1 \ldots E_n$ is encoded as the error tags (token identifier 22), the encoding of S_0, the sequence of the encodings of E_0 to E_n and the end error tags (token identifier 23).

Internal References

are encoded using the internal reference tags [30] and [30+128] (the sharing flag cannot be set on this tag, since chains of references are not allowed in the *OpenMath* binary encoding.) with long flag set if the number of *OpenMath* sub-objects in the encoded *OpenMath* is greater than or equal to 256. Then, there is the ordinal number of the referenced *OpenMath* object as a byte (if the long flag was not set) or a four byte integer (in network byte order).

External References

are encoded using the external reference tags [31] and [31+128] (the sharing flag cannot be set on this tag, since chains of references are not allowed in the *OpenMath* binary encoding) with the long flag set if the number of bytes in the reference URI is greater than or equal to 256. Then, there is the number of bytes in the URI used for the external reference as a byte (if the long flag was not set) or a four byte integer (in network byte order), followed by the URI.

4.2.2.1 Sharing in Objects begining with the identifier [24]

This binary encoding supports the sharing of symbols, variables and strings (up to a certain length for strings) within one object. That is, sharing between objects is not supported. A reference to a shared symbol, variable or string is encoded as the corresponding tag with the long flag not set and the shared flag set, followed by a positive integer n coded on one byte (0 to 255). This integer references the $n + 1$-th such sharable sub -object (symbol, variable or string up to 255 characters) in the current *OpenMath* object (counted in the order they are generated by the encoding). For example, `0x48 0x01` references a symbol that is identical to the second symbol that was found in the current object. Strings with 8 bit characters and strings with 16 bit characters are two different kinds of objects for this sharing. Only strings containing less than 256 characters can be shared (i.e. only strings up to 255 characters).

4.2.3 Sharing with References (beginning with [24+64])

In the binary encoding specified in the last section (which we keep for compatibility reasons, but deprecate in favor of the more efficient binary encoding specified in this section) only symbols, variables, and short strings could be shared. In this section, we will present a second binary encoding, which shares most of the identifiers with the one in the last one, but handles sharing differently. This encoding is signaled by the shared object tags [88].

The main difference is the interpretation of the sharing flag (bit 7), which can be set on all objects that allow it. Instead of encoding a reference to a previous occurrence of an object of the same type, it indicates whether an object will be referenced later in the encoding. This corresponds to the information, whether an `id` attribute is set in the xml encoding. On the object identifier (where sharing does not make sense), the shared flag signifies the encoding described here ([88]=[24+64]).

Otherwise integers, floats, variables, symbols, strings, bytearrays, and constructs are treated exactly as in the binary encoding described in the last section.

The binary encoding with references uses the additional reference tags [30] for (short) internal references, [30+128] for long internal references, [30] for (short) external references, [30+128] for long external references. Internal references are used to share sub-objects in the encoded object (see Figure 4.3 for an example) by referencing their position; external references allow to reference *OpenMath* objects in other documents by an URI.

Identifiers [30+64] and [30+64+128] are not used, since they would encode references that are shared themselves. Chains of references are redundant, and decrease both space and time efficiency, therefore they are not allowed in the *OpenMath* binary encoding.

References consist of the identifier [30] ([30+128] for long references) followed by a positive integer n coded on one byte (4 bytes for long references). This integer references the $n+1$th shared sub-object (one where the shared flag is set) in the current object (counted in the order they are generated in the encoding). For example `0x7E 0x01` references the second shared sub-object. Figure Figure 4.3 shows the binary encoding of the object in figure Figure 4.1 above.

Hex	Meaning	Hex	Meaning
58	begin object tag	05	variable tag
10	begin application tag	01	variable length
05	variable tag	61	a (variable name)
01	variable length	05	variable tag
66	f (variable name)	01	variable length
50	begin application tag (shared)	61	a (variable name)
05	variable tag	11	end application tag
01	variable length	1E	short reference
66	f (variable name)	00	to the first shared object
50	begin application tag (shared)	11	end application tag
05	variable tag	1E	short reference
01	variable length	00	to the second shared object
66	f (variable name)	11	end application tag
		19	end object tag

Figure 4.3 A binary encoding of the *OpenMath* object from figure Figure 4.1.

It is easy to see that in this binary encoding, the size of the encoding is $13+7(d-1)$ bytes, where d is the depth of the tree, while a totally unshared encoding is $8*2^d-8$ bytes (sharing variables saves up to 256 bytes for trees up to depth 8 and wastes space for greater depths). The shared xml encoding only uses $32d+29$ bytes, which is more space efficient starting at depth 9.

Note that in conversion to binary encoding the identifiers on the objects are not preserved. Morever, even though the xml encoding allows references across objects, as in figure ???, the binary encoding does not (the binary encoding has no notion of a multi-object collection, which is implicit by embedding *OpenMath* objects into xml documents in the xml encoding.).

Note that objects need not be fully shared (or shared at all) in the binary encoding with sharing.

4.2.4 Implementation Note

A typical implementation of the binary encoding comes in two parts. The first part deals with the unshared encodings, i.e. objects starting with the identifier [24].

This part uses four tables, each of 256 entries, for symbol, variables, 8 bit character strings whose lengths are less than 256 characters and 16 bit character strings whose lengths are less than 256 characters. When an object is read, all the tables are first flushed. Each time a sharable sub -object is read, it is entered in the corresponding table if it is not full. When a reference to the shared i-th object of a given type is read, it stands for the i-th entry in the corresponding table. It is an encoding error if the i-th position in the table has not already been assigned (i.e. forward references are not allowed). Sharing is not mandatory, there may be duplicate entries in the tables (if the application that wrote the object chose not to share optimally).

The part for the shared representations of *OpenMath* objects uses an unbounded array for storing shared sub-objects. Whenever an object has the shared flag set, then it is read and a pointer to the generated data structure is stored at the next position of the array. Whenever a reference of the form `[30] [_]` is encountered, the array is queried for the value at `[_]` and analogously for or `[30+128] {_}`. Note that the application can decide to copy the value or share it among subterms as long as it respects the identity conditions given by the tree -nature of the *OpenMath* objects. The implementation must take care to ensure that no variables are captured during this process (see section Section 4.1.2.2), and possibly have methods for recovering from cyclic dependency relations (this can be done by standard loop-checking methods).

Writing an object is simple. The tables are first flushed. Each time a sharable sub -object is encountered (in the natural order of output given by the encoding), it is either entered in the corresponding table (if it is not full) and output in the normal way or replaced by the right reference if it is already present in the table.

4.2.5 Example of Binary Encoding

As an example of this binary encoding, we can consider the *OpenMath* object whose xml encoding is

```
<OMOBJ>
  <OMA>
    <OMS uristub="http://www.openmath.org/cd" name="times" cd="arith1"/>
    <OMA>
      <OMS uristub="http://www.openmath.org/cd" name="plus" cd="arith1"/>
      <OMV name="x"/>
      <OMV name="y"/>
    </OMA>
    <OMA>
      <OMS uristub="http://www.openmath.org/cd" name="plus" cd="arith1"/>
      <OMV name="x"/>
      <OMV name="z"/>
    </OMA>
  </OMA>
</OMOBJ>
```

It is binary encoded as the sequence of bytes given by the following table.

Hex	Meaning	Hex		Meaning
18	begin object tag	68	h .)	
10	begin application tag	31	1 .)	
08	symbol tag	70	p (symbol name begin	
06	cd length	6c	l .	
05	name length	75	u .	
61	a (cd name begin	73	s .)	

72	r .	05	variable tag
69	i .	01	name length
74	t .	78	x (name)
68	h .	05	variable tag
31	1 .)	01	name length
74	t (symbol name begin	79	y (variable name)
69	i .	11	end application tag
6d	m .	10	begin application tag
65	e .	48	symbol tag (with share bit on)
73	s .)	01	reference to second symbol seen (arith1:plus)
10	begin application tag	45	variable tag (with share bit on)
08	symbol tag	00	reference to first variable seen (x)
06	cd length	05	variable tag
04	name length	01	name length
61	a (cd name begin	7a	z (variable name)
72	r .	11	end application tag
69	i .	11	end application tag
74	t .	19	end object tag

4.2.6 Relation to the *OpenMath*1 binary encoding

The *OpenMath*2 binary encoding significantly extends the *OpenMath*1 binary encoding to accomodate the new features and in particular sharing of sub-objects. The tags and structure of the *OpenMath*1 binary encoding are still present in the current *OpenMath* binary encoding, so that binary encoded *OpenMath*1 objects are still valid in the *OpenMath*2 binary encoding and correspond to the same abstract *OpenMath* objects. In some cases, the binary encoding tags without the shared flag can still be used as more compact representations of the objects (which are not shared, and do not have an identifier).

As the binary encoding is geared towards compactness, *OpenMath* objects should be maximally internally shared (if computationally feasible). Note that since sharing is done only at the encoding level, this does not alter the meaning of an *OpenMath* object, only allow to represent it more compactly.

4.3 Summary

The key points of this chapter are:

- The xml encoding for *OpenMath* objects uses most common character sets.

- The xml encoding is readable, writable and can be embedded in most documents and transport protocols.

- The binary encoding for *OpenMath* objects should be used when efficiency is a key issue. It is compact yet simple enough to allow fast encoding and decoding of objects.

Chapter 5
Content Dictionaries

In this chapter we give a brief overview of Content Dictionaries before explicitly stating their functionality and encoding.

5.1 Introduction

Content Dictionaries (CDs) are central to the *OpenMath* philosophy of transmitting mathematical information. It is the *OpenMath* Content Dictionaries which actually hold the meanings of the objects being transmitted.

For example if application *A* is talking to application *B*, and sends, say, an equation involving multiplication of matrices, then *A* and *B* must agree on what a matrix is, and on what matrix multiplication is, and even on what constitutes an equation. All this information is held within some Content Dictionaries which both applications agree upon.

A *Content Dictionary* holds the meanings of (various) mathematical "words". These words are *OpenMath* basic objects referred to as *symbols* in [Section 3.1](#).

With a set of symbol definitions (perhaps from several Content Dictionaries), *A* and *B* can now talk in a common "language".

It is important to stress that it is not Content Dictionaries themselves which are being transmitted, but some "mathematics" whose definitions are held within the Content Dictionaries. This means that the applications must have already agreed on a set of Content Dictionaries which they "understand" (i.e., can cope with to some degree).

In many cases, the Content Dictionaries that an application understands will be constant, and be intrinsic to the application's mathematical use. However the above approach can also be used for applications which can handle every Content Dictionary (such as an *OpenMath* parser, or perhaps a typesetting system), or alternatively for applications which understand a changeable number of Content Dictionaries (perhaps after being sent Content Dictionaries in some way).

The primary use of Content Dictionaries is thought to be for designers of Phrasebooks,the programs which translate between the *OpenMath* mathematical object and the corresponding (often internal) structure of the particular application in question. For such a use the Content Dictionaries have themselves been designed to be as readable and precise as possible.

Another possible use for *OpenMath* Content Dictionaries could rely on their automatic comprehension by a machine (e.g., when given definitions of objects defined in terms of previously understood ones), in which case Content Dictionaries may have to be passed as data. Towards this end, a Content Dictionary has been written which contains a set of symbols sufficient to represent any other Content Dictionary. This means that Content Dictionaries may be passed in the same way as other (*OpenMath*) mathematical data.

Finally, the syntax of the reference encoding for Content Dictionaries has been designed to be relatively easy to learn and to write, and also free from the need for any specialist software. This is because it is acknowledged that there is an enormous amount of mathematical information to represent, and so most of the Content Dictionaries will be written by "ordinary" mathematicians, encoding their particular fields of expertise. A further reason is that the mathematics conveyed by a specific Content Dictionary should be understandable independently of any application.

The key points from this section are:

- Content Dictionaries should be readable and precise to help Phrasebook designers,

- Content Dictionaries should be readily write-able to encourage widespread use,

- It ought to be possible for a machine to understand a Content Dictionary to some degree.

5.2 Abstract Content Dictionaries

In this section we define the ~~overall~~abstract structure of Content Dictionaries.

~~Other than Content Dictionary comments (which have no real semantics), Content Dictionaries have been designed to hold two types of information: that which is pertinent to the whole Content Dictionary, and that which is restricted to a particular symbol definition. Specific information pertaining to the symbols like the signature and the defining mathematical properties is conveyed in additional files associated to Content Dictionaries.~~

~~Information that is pertinent to the whole Content Dictionary includes:~~

- ~~The name of the Content Dictionary.~~

- ~~A description of the Content Dictionary.~~

- ~~A date when the Content Dictionary is next planned to be reviewed.~~

- ~~A date on which the Content Dictionary was last edited.~~

- ~~The current version and revision numbers of the Content Dictionary.~~

- ~~The status of the Content Dictionary.~~

- ~~An optional URL for this Content Dictionary.~~

- ~~An optional list of Content Dictionaries on which this Content Dictionary depends. That is, those named in Examples and FMP in this Content Dictionary.~~

- ~~An optional comment, possibly containing the author's name.~~

~~Information that is restricted to a particular symbol includes:~~

- ~~The name of the symbol.~~

- ~~A description of this symbol.~~

- ~~An optional comment.~~

- ~~Optional properties that this symbol should obey.~~

- ~~Optional examples of the use of this symbol.~~

A Content Dictionary consists of the following mandatory pieces of information:

1. A *name* corresponding to the rules described in [Section 3.3](#).

2. A *description* of the Content Dictionary.

3. A *revision date*, the date of the last change to the Content Dictionary. Dates should be stored in the ISO-compliant format YYYY-MM-DD, e.g. 1966-02-03.

4. A *review date*, a date until which the content dictionary is guarenteed to remain unchanged.

5. A *version number* which consists of a major and minor part (see Section 5.2.2).

6. A *status*, as described in Section 5.2.1.

7. A *uri stub* which is a unique identifier for the Content Dictionary, and may or may not refer to an actual location from which it can be retrieved.

8. A series of *symbol definitions* as described below.

A symbol definition consists of the following pieces of information:

1. A mandatory *name* corresponding to the rules described in Section 3.3.

2. A mandatory *description* of the symbol, which can be as formal or informal as the author likes.

3. An optional *role* as described in Section 3.1.4.

4. Zero or more *commented mathematical properties* which are mathematical properties of the symbol expressed in a mechanism other than *OpenMath*.

5. Zero or more *formal mathematical properties* which are mathematical properties of the symbol expressed in *OpenMath*. Note that it is common for commented and formal mathematical properties to be introduced in pairs, with the former describing the latter.

6. Zero or more *examples* which are intended to demonstrate the use of the symbol within an *OpenMath* object.

~~As mentioned earlier, certain kinds of data pertaining to symbols may be conveyed in files other than a Content Dictionary. In particular, information on signatures according to a type system may be described in *Signature Files* whose format is given in Section 5.4.1. Other information such as presentation forms, extra defining mathematical properties may be associated with Content Dictionaries using files whose format is not specified by this standard. It is expected that a common method of defining the presentation for *OpenMath* symbols is via xsl [20] stylesheets giving transformations to MathML.~~

Some pieces of information which might logically be thought to be part of a Content Dictionary, such as the types or signatures of symbols, are better represented externally. This allows for new varients to be associated with Content Dictionaries without the Dictionaries themselves being changed. A model for signatures is given in Section 5.4.1.

Content Dictionaries may be grouped into *CD Groups*. These groups allow applications to easily refer to collections of Content Dictionaries. One particular CDGroup of interest is the "MathML CDGroup". This group expresses the collection of the core Content Dictionaries that is designed to have the same semantic scope as the content elements of MathML 2 [19]. *OpenMath* objects built from symbols that come from Content Dictionaries in this CDGroup may be expected to be eaily transformed between *OpenMath* and MathML encodings. The detailed structure of a CDGroup is described in Section 5.4.2 below.

5.2.1 Content Dictionary Status

The status of a Content Dictionary can be either

- `official`: approved by the *OpenMath* Society according to the procedure outlined in Section 5.5;

- `experimental`: under development and thus liable to change;

- `private`: used by a private group of *OpenMath* users;

- `obsolete`: an obsolete Content Dictionary kept only for archival purposes.

5.2.2 Content Dictionary Version Numbers

A version number must consist of two parts, a major version and a revision, both of which should be non-negative integers. In CDs that do not have status *experimental*, the version number should be a positive integer.

Unless a CD has status *experimental*, no changes should ever be introduced that invalidate objects built with previous versions. Any change that influences phrasebook compliance, like adding a new symbol to a Content Dictionary, is considered a major change and should be reflected by an increase in the major version number. Other changes, like adding an example or correcting a description, are considered minor changes. For minor changes the version number is not changed, but an increase should be made to the revision number. Note that a change such as removing a symbol should not be made unless the CD has status *experimental*. Should this be required then a new CD with a different name should be produced so as not to invalidate existing objects.

When the major version number is increased, the revision number is normally reset to zero.

As detailed in chapter Chapter 6, *OpenMath* compliant applications state which versions of which CDs they support.

5.3 The ~~xml~~Reference Encoding for Content Dictionaries

The reference encoding of Content Dictionaries are as xml documents. A valid Content Dictionary document should

- ~~be valid according to the DTD given in Figure 5.1,~~

- ~~adhere to the extra conditions on the content of the elements given in Section 5.3.3.~~

conform to the Relax NG Schema for Content Dictionaries given in Section 5.3.1.

An example of a complete Content Dictionary is given in Appendix Appendix A.1, which is the `Meta` Content Dictionary for describing Content Dictionaries themselves. A more typical Content Dictionary is given in Appendix Appendix A.2, the `arith1` Content Dictionary for basic arithmetic functions.

5.3.1 The RelaxNG Schema for Content Dictionaries

```
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<!-- ******************************************** -->
<!--                                              -->
<!--  Relax NG Schema for OpenMath CD            -->
<!--                                              -->
<!-- ******************************************** -->
<grammar ns="http://www.openmath.org/OpenMath"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <define name="CDComment">
    <element name="CDComment">
      <ref name="attlist.CDComment"/>
      <text/>
    </element>
  </define>
  <define name="attlist.CDComment" combine="interleave">
    <empty/>
  </define>
  <define name="CDName">
    <element name="CDName">
      <ref name="attlist.CDName"/>
      <text/>
    </element>
  </define>
  <define name="attlist.CDName" combine="interleave">
    <empty/>
  </define>
  <define name="CDURL">
    <element name="CDURL">
      <ref name="attlist.CDURL"/>
      <text/>
    </element>
  </define>
  <define name="attlist.CDURL" combine="interleave">
    <empty/>
  </define>
  <define name="CDURISTUB">
    <element name="CDURISTUB">
      <ref name="attlist.CDURISTUB"/>
      <text/>
    </element>
  </define>
  <define name="attlist.CDURISTUB" combine="interleave">
    <empty/>
  </define>
  <define name="CDUses">
    <element name="CDUses">
      <ref name="attlist.CDUses"/>
      <zeroOrMore>
        <ref name="CDName"/>
      </zeroOrMore>
    </element>
  </define>
  <define name="attlist.CDUses" combine="interleave">
    <empty/>
  </define>
  <define name="inhel">
    <text/>
  </define>
  <define name="inhel2">
    <zeroOrMore>
      <choice>
        <text/>
        <ref name="OMOBJ"/>
      </choice>
    </zeroOrMore>
  </define>
  <define name="CDReviewDate">
    <element name="CDReviewDate">
      <ref name="attlist.CDReviewDate"/>
      <ref name="inhel"/>
    </element>
  </define>
```

```xml
<define name="attlist.CDReviewDate" combine="interleave">
  <empty/>
</define>
<define name="CDStatus">
  <element name="CDStatus">
    <ref name="attlist.CDStatus"/>
    <ref name="inhel"/>
  </element>
</define>
<define name="attlist.CDStatus" combine="interleave">
  <empty/>
</define>
<define name="Description">
  <element name="Description">
    <ref name="attlist.Description"/>
    <ref name="inhel"/>
  </element>
</define>
<define name="attlist.Description" combine="interleave">
  <empty/>
</define>
<define name="Name">
  <element name="Name">
    <ref name="attlist.Name"/>
    <ref name="inhel"/>
  </element>
</define>
<define name="attlist.Name" combine="interleave">
  <empty/>
</define>
<define name="Role">
  <element name="Role">
    <ref name="attlist.Role"/>
    <ref name="inhel"/>
  </element>
</define>
<define name="attlist.Role" combine="interleave">
  <empty/>
</define>
<define name="CMP">
  <element name="CMP">
    <ref name="attlist.CMP"/>
    <ref name="inhel"/>
  </element>
</define>
<define name="attlist.CMP" combine="interleave">
  <empty/>
</define>
<include href="openmath.rng"/>
<define name="FMP">
  <element name="FMP">
    <ref name="attlist.FMP"/>
    <ref name="inhel2"/>
  </element>
</define>
<define name="attlist.FMP" combine="interleave">
  <empty/>
</define>
<!-- allow embedded OM -->
<define name="Example">
  <element name="Example">
    <ref name="attlist.Example"/>
    <ref name="inhel2"/>
  </element>
</define>
<define name="attlist.Example" combine="interleave">
  <empty/>
</define>
<define name="CDDefinition">
  <element name="CDDefinition">
    <ref name="attlist.CDDefinition"/>
```

```
        <zeroOrMore>
          <choice>
            <ref name="CDComment"/>
            <ref name="Name"/>
            <ref name="Description"/>
            <ref name="Example"/>
            <ref name="FMP"/>
            <ref name="CMP"/>
          </choice>
        </zeroOrMore>
      </element>
  </define>
  <define name="attlist.CDDefinition" combine="interleave">
    <empty/>
  </define>
  <define name="CD">
    <element name="CD">
        <ref name="attlist.CD"/>
        <zeroOrMore>
          <choice>
            <ref name="CDComment"/>
            <ref name="CDName"/>
            <ref name="Description"/>
            <ref name="CDReviewDate"/>
            <ref name="CDStatus"/>
            <ref name="CDURL"/>
            <optional>
              <ref name="CDURI"/>
            </optional>
            <optional>
              <ref name="CDUses"/>
            </optional>
            <ref name="CDDefinition"/>
            <ref name="Example"/>
          </choice>
        </zeroOrMore>
      </element>
  </define>
  <define name="attlist.CD" combine="interleave">
    <empty/>
  </define>
  <define name="CDURI" combine="choice">
    <notAllowed/>
  </define>
  <define name="OMOBJ" combine="choice">
    <notAllowed/>
  </define>
  <start>
    <choice>
      <ref name="CD"/>
      <ref name="Role"/>
      <ref name="CDURISTUB"/>
    </choice>
  </start>
</grammar>
```

5.3.2 The DTD Specification of Content Dictionaries

```
<!-- DTD for OpenMath object  sb 29.10.1998 -->

<!-- general list of embeddable elements
     · excludes OMATP as this is only embeddable in OMATTR
     · excludes OMBVAR as this is only embeddable in OMBIND
-->
<!ENTITY % omel "OMS | OMV | OMI | OMB | OMSTR
                     | OMF | OMA | OMBIND | OME  | OMATTR  ">
```

```
<!-- things which can be variables -->
<!ENTITY % omvar        "OMV | OMATTR" >

<!-- symbol -->
<!ELEMENT OMS EMPTY>
<!ATTLIST OMS name CDATA #REQUIRED  cd CDATA #REQUIRED >

<!-- variable -->
<!ELEMENT OMV EMPTY>
<!ATTLIST OMV name CDATA #REQUIRED >

<!-- integer -->
<!ELEMENT OMI (#PCDATA) >

<!-- byte array -->
<!ELEMENT OMB (#PCDATA) >

<!-- string -->
<!ELEMENT OMSTR (#PCDATA) >

<!-- floating point -->
<!ELEMENT OMF EMPTY>
<!ATTLIST OMF dec CDATA #IMPLIED  hex CDATA #IMPLIED>

<!-- apply constructor -->
<!ELEMENT OMA (%omel;)+ >

<!-- binding constructor & variable -->
<!ELEMENT OMBIND ((%omel;), OMBVAR, (%omel;)) >
<!ELEMENT OMBVAR (%omvar;)+ >

<!-- error -->
<!ELEMENT OME    (OMS, (%omel;)* ) >

<!-- attribution constructor & attribute pair constructor -->
<!ELEMENT OMATTR (OMATP, (%omel;)) >
<!ELEMENT OMATP (OMS, (%omel;))+ >

<!-- OM object constructor -->
<!ELEMENT OMOBJ (%omel;) >

<!-- end of DTD for OM object -->
```

Figure 5.1 DTD Specification of Content Dictionaries

The xml DTD for Content Dictionaries is given in   Figure 5.1. The allowed elements are further described in the following section.

5.3.3 Further ~~Requirements of an *OpenMath* Content Dictionary~~ Description of the CD Schema

~~The notion of being a valid Content Dictionary is stronger than merely being successfully parsed by the DTD. This is because the content of the elements, referred to in   Figure 5.1 as PCDATA and CDATA, must actually make sense to, say, a Phrasebook designer. In this section we define exactly the format of the elements used in Content Dictionaries.~~

We now describe the elements used in the above schema in terms of the abstract description of CDs in Section 5.2. Unless stated otherwise the information is encoded as the content of the element.

CDName

The ~~text occurring in the~~ `CDName` ~~element corresponds to the~~ name of the Content Dictionary.

Description

The text occurring in the `Description` element is used to give a description of the enclosing element, which could be a symbol or the entire Content Dictionary. The content of this element can be any xml text.

CDReviewDate

The ~~text occurring in the~~ `CDReviewDate` ~~element corresponds to the earliest possible~~ review date of the Content Dictionary. ~~The date formats should be ISO-compliant in the form YYYY-MM-DD, e.g. 1953-09-26.~~

CDDate

The ~~text occurring in the~~ `CDDate` ~~element corresponds to the~~ revision date of this version of the Content Dictionary. ~~The date formats should be ISO-compliant in the form YYYY-MM-DD, e.g. 1953-09-26.~~

CDVersion

The major version number of the CD.

~~The text occurring in the~~ `CDVersion` ~~element corresponds to the version number of the current version of a Content Dictionary. It should be a non-negative integer.~~

~~In CDs that do not have status~~ *experimental*~~, CD version numbering should adhere to the following. The version number should be a positive integer.~~

~~No changes can be introduced that invalidate objects built with previous versions. Any change that influences phrasebook compliance, like adding a new symbol to a Content Dictionary, is considered a major change. and should be reflected by an increase in this version number. Other changes, like adding an example or correcting a description, are considered minor changes. For minor changes the version number is not changed, but an increas should be made to the revision number, as described below. A change such as removing a symbol should not be made, instead a new CD, with a different name should be produced, so as not to invalidate existing objects.~~

~~As detailed in chapter~~ ~~Chapter 6,~~ *OpenMath* ~~compliant applications state which versions of which CDs they support.~~ *Experimental* ~~CDs may expect to have changes such as adding or removing symbols as they are developed, without requiring the name of the CD to be changed.~~

CDRevision

~~The text occurring in the~~ `CDRevision` ~~element corresponds to the revision, or `minor version number' of the current version of a Content Dictionary. It should be a non-negative integer.~~

~~Minor changes to a CD that do not warrant the release of a CD with an increased version~~

number should be marked by increasing the revision number specified in this field. When the Cd Version number is increased, the Revision number is normally reset to zero.

The minor version number of the CD.

CDStatus

The text occurring in the CDStatus element corresponds to the status of the Content Dictionary. , and can be either official (approved by the *OpenMath* Society according to the procedure outlined in Section 5.5), experimental (currently being tested), private (used by a private group of *OpenMath* users) or obsolete (an obsolete Content Dictionary kept only for archival purposes).

CDURISTUB

The uri stub of the CD.

CDURL

The text occurring in the CDURL element should be a valid URL where the source file for the Content Dictionary encoding can be found (if it exists). The filename should conform to ISO 9660 [14].

CDUses

The content of this element should be a series of CDName elements, each naming a Content Dictionary used in the Example and FMPs of the current Content Dictionary. This element is optional and deprecated since the information can easily be extracted automatically.

CDComment

The content of this element should be text that does not convey any crucial information concerning the current Content Dictionary. It can be used in the Content Dictionary header to report the author of the Content Dictionary and to log change information. In the body of the Content Dictionary, it can be used to attach extra remarks to certain symbols.

CDDefinition

The element which contains the definition of an individual symbol.

Name

The text occurring in the Name element corresponds to the name of a symbol. , and is specified as in Chapter 4.

Example

The text occurring in the Example element is used to give examples of the enclosing symbol, and can be any xml text. In addition to text the element may contain examples as xml

encoded *OpenMath*, inside `OMOBJ` elements. Note that `Examples` must be with respect to some symbol and cannot be "loose" in the Content Dictionary.

`CMP`

~~The text occurring in the CMP element corresponds to a property of the symbol. An application which says it understands a Content Dictionary symbol need not understand a commented property of the symbol.~~ A Commented Mathematical Property.

`FMP`

~~The content of the FMP element also corresponds to a property *1 of the symbol, however the content of this element must be a valid OpenMath object in the xml encoding. An application which says it understands a Content Dictionary symbol need not understand a formal property of the symbol.~~ A Formal Mathematical Property.

5.4 Additional Information

Content Dictionaries contain just one part of the information that can be associated to a symbol in order to stepwise define its meaning and its functionality. *OpenMath* Signature ~~files~~ dictionaries, CDGroups, and possibly ~~files~~ collections of extra mathematical properties, are used to convey the different aspects that as a whole make up a mathematical definition.

5.4.1 Signature ~~Files~~ Dictionaries

OpenMath may be used with any type system. One just needs to produce a Content Dictionary which gives the constructors of the type system, and then one may build *OpenMath* objects representing types in the given type system. These are typically associated with *OpenMath* objects via the *OpenMath* **attribution** constructor.

A Small Type System, called STS, has been designed to give semi-formal signatures to *OpenMath* symbols and is documented in [6]. The signature file given in Appendix A.3 is based on this formalism. Using the same mechanism, [4] shows how pure type systems can also be employed to assign types to *OpenMath* symbols.

~~5.4.1.1 The DTD Specification of Signature Files~~

~~Signature Files are xml documents, hence a valid Signature File should~~

- ~~be valid according to the dtd given in Figure 5.2,~~

- ~~adhere to the extra conditions on the content of the elements given in Section 5.4.1.2.~~

~~Signature files have a header which specifies the Content Dictionary and determines the type system being used, and the Content Dictionary which contains the symbols for which the signatures are being given. Each signature takes the form of an xml encoded OpenMath object.~~

```
<!-- omcdsig.dtd -->
<!-- ************************************** -->
<!-- -->
<!-- DTD for OpenMath CD Signatures -->
<!-- (c) EP24969 the ESPRIT OpenMath Consortium -->
```

```
<!-- David Carlisle 1999-04-13                            -->
<!-- David Carlisle 1999-05-21                            -->
<!-- David Carlisle 1999-06-22                            -->
<!--                                                      -->
<!--                                                      -->
<!-- ****************************************** -->

<!-- include dtd for OM objects -->
<!ENTITY % omobjectdtd SYSTEM "omobj.dtd" >
%omobjectdtd;

<!ELEMENT CDSComment       (#PCDATA) >
<!ELEMENT CDSReviewDate    (#PCDATA) >
<!ELEMENT CDSStatus      (#PCDATA) >

<!ELEMENT CDSignatures   (CDComment |CDSComment | CDSReviewDate |
                          CDSStatus | Signature )* >

<!ATTLIST CDSignatures cd CDATA #REQUIRED
                       type CDATA #REQUIRED >

<!ELEMENT Signature       (OMOBJ?) >

<!ATTLIST  Signature  name CDATA #REQUIRED >

<!-- end of DTD for OM CD Signatures -->
```

Figure 5.2 DTD Specification of Signature Files

5.4.1.2 Further Requirements Abstract Specification of a Signature File

Signature files have a header which specifies the type system being used, and the Content Dictionary which contains the symbols for which the signatures are being given. Each signature takes the form of an *OpenMath* object in an appropriate encoding.

The notion of being a valid Signature File is stronger than merely being successfully parsed by the dtd in Figure 5.2. In this section we define exactly the format of the elements used in Signature Files. Several of the requirements are the same as those on elements of Contents Dictionaries.

CDSignatures

The outermost element of the Signature File is characterized by two required attributes that identify the type system and the Content Dictionary whose signatures are defined. The value of the xml attribute `type` is the name of the Content Dictionary or of the CDGroup (cfg. Section 5.4.2) that represents the type system. The value of the xml attribute `cd` is the name of the Content Dictionary whose symbols are assigned signatures in this Signature File. Both values are of the form specified in Chapter 4.

CDSComment

See `CDComment` in Section 5.3.3.

CDSreviewDate

The text occurring in the `CDSReviewDate` element corresponds to the earliest possible revision date of the Signature File. The date formats should be ISO compliant in the form YYYY-MM-DD, e.g. 2000-02-29.

1. A *type definition*: the name of the Content Dictionary or of the CDGroup (cfg. Section 5.4.2) that represents the type system being used.

2. A *CD name*: the name of the CD for which signatures are being defined.

3. A *review date* as defined in Section 5.2.

4. A *status*: as defined in Section 5.2.

5. A series of *signatures* which are *OpenMath* objects in some encoding. The objects must represent types as defined by the type definition.

5.4.1.3 A RelaxNG Schema for a Signature File

The following is a reference encoding of a signature dictionary, designed to be used with Content Dictionaries in the xml encoding.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ***************************************** -->
<!--                                           -->
<!-- Relax NG Schema for OpenMath CD Signatures   -->
<!--                                           -->
<!-- ***************************************** -->
<grammar ns="http://www.openmath.org/OpenMath"
         xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="openmath.rng"/>
  <define name="CDSComment">
    <element name="CDSComment">
      <ref name="attlist.CDSComment"/>
      <text/>
    </element>
  </define>
  <define name="attlist.CDSComment" combine="interleave">
    <empty/>
  </define>
  <define name="CDSReviewDate">
    <element name="CDSReviewDate">
      <ref name="attlist.CDSReviewDate"/>
      <text/>
    </element>
  </define>
```

```
      <define name="attlist.CDSReviewDate" combine="interleave">
        <empty/>
      </define>
      <define name="CDSStatus">
        <element name="CDSStatus">
          <ref name="attlist.CDSStatus"/>
          <text/>
        </element>
      </define>
      <define name="attlist.CDSStatus" combine="interleave">
        <empty/>
      </define>
      <define name="CDSignatures">
        <element name="CDSignatures">
          <ref name="attlist.CDSignatures"/>
          <zeroOrMore>
            <choice>
              <ref name="CDComment"/>
              <ref name="CDSComment"/>
              <ref name="CDSReviewDate"/>
              <ref name="CDSStatus"/>
              <ref name="Signature"/>
            </choice>
          </zeroOrMore>
        </element>
      </define>
      <define name="attlist.CDSignatures" combine="interleave">
        <attribute name="cd"/>
        <attribute name="type"/>
      </define>
      <define name="Signature">
        <element name="Signature">
          <ref name="attlist.Signature"/>
          <optional>
            <ref name="OMOBJ"/>
          </optional>
        </element>
      </define>
      <define name="attlist.Signature" combine="interleave">
        <attribute name="name"/>
      </define>
      <define name="CDComment" combine="choice">
        <notAllowed/>
      </define>
      <define name="OMOBJ" combine="choice">
        <notAllowed/>
      </define>
      <start>
        <choice>
          <ref name="CDSignatures"/>
        </choice>
      </start>
   </grammar>
```

5.4.1.4 Examples

An example of a signature file for the type system STS and the `arith1` Content Dictionary is given in [Appendix A.3](#). Each signature entry is similar to the following one for the *OpenMath* symbol `<OMS cd="arith1" name="plus"/>`:

```
<Signature name="plus">
<OMOBJ>
 <OMA>
  <OMS uristub="http://www.openmath.org/cd" name="mapsto" cd="sts"/>
  <OMA>
```

```
    <OMS uristub="http://www.openmath.org/cd" name="nassoc" cd="sts"/>
    <OMV name="AbelianSemiGroup"/>
  </OMA>
  <OMV name="AbelianSemiGroup"/>
 </OMA>
</OMOBJ>
</Signature>
```

5.4.2 CDGroups

The CD Group mechanism is a convenience mechanism for identifying collections of CDs. A CD Group file is an xml document used in the (static or dynamic) negotiation phase where communicating applications declare and agree on the Content Dictionaries which they process. It is a complement, or an alternative, to the individual declaration of Content Dictionaries understood by an application. Note that CD Groups do *not* affect the *OpenMath* objects themselves. Symbols in an object always refer to content dictionaries, not groups.

For an application to declare that it "understands CDGroup G" is exactly equivalent to, and interchangable with, the declaration that it "understands Content Dictionaries $x_1, x_2, \ldots x_n$", where x_1, $\ldots x_n$ are the members of CDGroup G.

5.4.2.1 The DTD Specification of CDGroups

CDGroups are xml documents, hence a valid CDGroup should

- be valid according to the DTD given in Figure 5.3,

- adhere to the extra conditions on the content of the elements given in Section 5.4.2.2.

Apart from some header information such as CDGroupName and CDGroup version, a CDGroup is simply an unordered list of CDs, identified by name and optionally version number and URL.

```
<!-- CDgroup.dtd -->
<!-- ******************************************** -->
<!--                                              -->
<!-- DTD for OpenMath CD group                    -->
<!-- (c) EP24969 the ESPRIT OpenMath Consortium   -->
<!-- date = 18.Feb.1999                           -->
<!-- author = s.buswell sb@stilo.demon.co.uk      -->
<!--                                              -->
<!--                                              -->
<!-- available at                                 -->
<!-- http://www.nag.co.uk/~something here David~  -->
<!--                                              -->
<!-- ******************************************** -->

<!-- info on the CD group itself -->

<!ELEMENT CDGroupName        (#PCDATA) >
<!ELEMENT CDGroupVersion     (#PCDATA) >
<!ELEMENT CDGroupRevision    (#PCDATA) >
<!ELEMENT CDGroupURL         (#PCDATA) >
<!ELEMENT CDGroupDescription (#PCDATA) >

<!-- info on the CDs in the group  -->

<!ELEMENT CDComment      (#PCDATA) >
```

```
<!ELEMENT CDGroupMember (CDComment?,CDName, CDVersion?, CDURL?) >
<!ELEMENT CDName      (#PCDATA) >
<!ELEMENT CDVersion    (#PCDATA) >
<!ELEMENT CDURL        (#PCDATA) >

<!-- structure of the group -->
<!ELEMENT CDGroup
    (CDGroupName, CDGroupVersion, CDGroupRevision?,
     CDGroupURL, CDGroupDescription,
      (CDGroupMember  | CDComment )* ) >

<!-- end of DTD for OM CDGroup -->
```

Figure 5.3 DTD Specification of CDGroups

5.4.2.2 Further Requirements of a CDGroup

The notion of being a valid CDGroup implies that the following requirements on the content of the elements described by the DTD in Figure 5.2 are also met.

CDGroup

> The xml element CDGroup is the outermost element in a CDGroup document.

CDGroupName

> The text occurring in the   CDGroupName element corresponds to the name of the CDGroup. For the syntactical requirements, see CDName in Section 5.3.3.

CDGroupURL

> The text occurring in the CDGroupURL element identifies the location of the CDGroup file, not necessarily of the member Content Dictionaries. For the syntactical requirements, see CDURL in Section 5.3.3.

CDGroupDescription

> The text occurring in the CDGroupDescription element describes the mathematical area of the CDGroup.

CDGroupMember

> The xml element CDGroupMember encloses the data identifying each member of the CDGroup.

CDName

> The text occurring in the CDName element corresponds to the name of a Content Dictionary in the CDGroup. For the syntactical requirements, see CDName in Section 5.3.3.

CDVersion

The text occurring in the `CDVersion` element identifies which version of the Content Dictionary isto be taken as member of the CDGroup. This element is optional. In case it is missing, the latest version is the one included in the CDGroup. For the syntactical requirements, see `CDVersion` in Section 5.3.3.

`CDURL`

The text occurring in the `CDURL` element identifies the location of the Content Dictionary to be taken as member of the CDGroup. This element is optional. In case it is missing, the location of the CDGroup identified by the element `CDGroupURL` is assumed. For the syntactical requirements, see `CDURL` in Section 5.3.3.

`CDComment`

See `CDComment` in Section 5.3.3.

5.5 Content Dictionaries Reviewing Process

The *OpenMath* Society is responsible for implementing a review and referee process to assess the accuracy of the mathematical content of Content Dictionaries. The status (see `CDStatus`) and/or the version number (see `CDVersion`) of a Content Dictionary may change as a result of this review process.

Chapter 6
OpenMath Compliance

Applications that meet the requirements specified in this chapter may label themselves as *OpenMath compliant*. *OpenMath* compliancy is defined so as to maximize the potential for interoperability amongst *OpenMath* applications.

6.1 Encoding

This standard defines two reference encodings for *OpenMath*, the binary encoding and xml encoding, defined in Chapter 4.

As a minimum, an *OpenMath* compliant application, which accepts or generates *OpenMath* objects, *must* be capable of doing so using the xml encoding. The ability to use other encodings is optional.

6.2 Content Dictionaries

An *OpenMath* compliant application *must* be able to support the error Content Dictionary defined in Appendix A.5.

A compliant application must declare the names and version numbers of the Content Dictionaries that it supports. Equivalently it may declare the Content Dictionary Group (or groups) and major version number (not revision number), rather than listing individual Content Dictionaries. Applications that support all Content Dictionaries (e.g. renderers) should refer to the implicit CD Group `all`.

If a compliant application supports a Content Dictionary then it must explicitly declare any symbols in the Content Dictionaries that are not supported. Phrasebooks are encouraged to support every symbol in the Content Dictionaries.

Symbols which are not listed as unsupported are *supported* by the application. The meaning of *supported* will depend on the application domain. For example an *OpenMath* renderer should provide a default display for any *OpenMath* object that only references supported symbols, whereas a Computer Algebra System will be expected to map such an object to a suitable internal representation, in this system, of this mathematical object. It is expected that the application's *phrasebooks* for supported Content Dictionaries will be constructed such that propertes of the symbol expressed in the Content Dictionary are respected as far as possible for the given application domain. However *OpenMath* compliance does *not* imply any guarantee by the *OpenMath* Society on the accuracy of these representations.

Content Dictionaries available from the official *OpenMath* repository at www.openmath.org need only be referenced by name, other Content Dictionaries *should* be referenced ~~by the URL declared in the CDURL field of the Dictionary. This URL may be used to retrieve the Content Dictionary.~~ using the `URISTUB` and the `CDName`.

When receiving an *OpenMath* symbol, e.g. *s*, that is not supported from a supported Content Dictionary, a compliant application will act as if it had received the *OpenMath* object

$$\mathbf{error}(\text{Unhandled\_Symbol}, s)$$

where `Unhandled_Symbol` is the symbol from the error Content Dictionary.

Similarly if it receives a symbol, e.g. *s*, from an unsupported Content Dictionary, it will act as if it had received the *OpenMath* object

$$\mathbf{error}(\text{Unsupported\_CD}, s)$$

Finally if the compliant application receives a symbol from a supported Content Dictionary but with an unknown name, then this must either be an incorrect object, or possibly the object has been built using a later version of the Content Dictionary. In either case, the application will act as if it had received the *OpenMath* object

$$\mathbf{error}(\text{Unexpected\_Symbol}, s)$$

6.3 Lexical Errors

The previous section defines the behaviour of a compliant application upon receiving well formed *OpenMath* objects containing unexpected symbols. This standard does not specify any behaviour for an application upon receiving ill-formed objects.

Chapter 7
Conclusion

The goal of this document is to define the *OpenMath* standard. The things are addressed by the *OpenMath* standard are:

- Informal and formal definition of the *OpenMath* objects.

- Informal and formal definition of the notion of Content Dictionaries.

To do this, *OpenMath* objects are precisely defined and two encodings are described to represent these objects using xml and binary code. Furthermore, the Document Type Definition for validating

Content Dictionaries and *OpenMath* objects is given.

Appendix A
CD Files

A.1 The meta Content Dictionary

```
<CD>

  <CDName> meta </CDName>

  <Description>

    This is a content dictionary to represent content dictionaries, so
    that they may be passed between &OM; compliant application in a
    similar way to mathematical objects.  It is acknowledged that this is
    not the only way to do this, but it seems a natural way.

    This can be viewed as updating the previous Meta-CD.

    The information written here is taken from "The &OM; Standard".
    This document is a slightly stronger statement than "the following
    symbols are defined as in the DTD at
    http://www.nag.co.uk/projects/OpenMath/omstd/dtds/cd.dtd", since the
    DTD often only says that the information inside the elements is PCDATA
    without saying what this actually corresponds to. However this is the
    only way this document is better than the DTD, and thus this is the
    only extra information we give here.

    Author: N. Howgrave-Graham
  </Description>

  <CDReviewDate> 1998-10-01 </CDReviewDate>
  <CDStatus> experimental </CDStatus>
  <CDURL> http://www.nag.co.uk/projects/OpenMath/omstd/cds/meta.ocd </CDURL>

  <CDComment>
    This is how one uses comments.

    This whole document must be valid &exml; which means that we cannot have
    sub-elements inside elements where we are expecting PCDATA. For this
    reason it is suggested that one use the unicode characters when
    clashes occur. For example &lt; and &gt; for less than and more than.
  </CDComment>

  <CDDefinition>
    <Name> CD </Name>
    <Description>
      The DTD fully specifies the use of the CD tag
    </Description>
  </CDDefinition>

  <CDDefinition>
    <Name> CDDescription </Name>
    <Description>
      The DTD fully specifies the use of the CDDescription tag
    </Description>
  </CDDefinition>

  <CDComment>
    For those that do not have access to the DTD, the tagged entries
    are the following (in no particular order):

    &lt;CD&gt;
    &lt;CDName&gt; &lt;/CDName&gt;
```

```
        &lt;Description&gt; &lt;/Description&gt;
        &lt;CDReviewDate&gt; &lt;/CDReviewDate&gt;
        &lt;CDStatus&gt; &lt;/CDStatus&gt;
        &lt;CDURL&gt;? &lt;/CDURL&gt;
        &lt;CDUses&gt;? &lt;CDUses&gt;
        &lt;CDDefinition&gt;*
        &lt;Name&gt; &lt;/Name&gt;
        &lt;Description&gt; &lt;/Description&gt;
        &lt;Signature&gt;? &lt;/Signature&gt;
        &lt;Example&gt;* &lt;/Example&gt;
        &lt;FMP&gt;* &lt;/FMP&gt;
        &lt;CMP&gt;* &lt;/CMP&gt;
        &lt;Presentation&gt;? &lt;/Presentation&gt;
        &lt;/CDDefinition&gt;

      where an asterisk (?) denotes it can repeated 0 or 1 times, and a star
      (*) denotes 0 or more times.
    </CDComment>

    <CDDefinition>
      <Name> CDName </Name>
      <Description>
        A tag which contains PCDATA corresponding to the name of the CD.
      </Description>
    </CDDefinition>

    <CDDefinition>
      <Name> CDURL </Name>
      <Description>
        An optional tag which contains PCDATA corresponding to the URL where
        the CD is stored.
      </Description>
    </CDDefinition>

    <CDDefinition>
      <Name> Example </Name>
      <Description>
        A tag which contains PCDATA to give an example of the
        enclosing symbol definition.
      </Description>
    </CDDefinition>

    <CDDefinition>
      <Name> CDReviewDate </Name>
      <Description>
        A tag which contains PCDATA to give an expiry date of
        the CD.  It should be in the form of ISO-8601, i.e. YYYY-MM-DD
      </Description>
    </CDDefinition>

    <CDDefinition>
      <Name> CDStatus </Name>
      <Description>
        A tag which contains PCDATA to give information on the
        status of the CD. This can be either official (approved by the
        &OM; steering committee), experimental (currently being tested),
        private (used by a private group of &OM; users) or obsolete
        (an obsolete CD kept only for archival purposes).
      </Description>
    </CDDefinition>

    <CDDefinition>
      <Name> CDUses </Name>
      <Description>
        A tag which contains zero or more CDNames which correspond
        to the CD's that this CD depends on. This makes an inheritance
        structure for CD's. If the CD is dependent on any other CD's they must
be present here.
</Description>
</CDDefinition>
```

```
<CDDefinition>
  <Name> CDTypeUses </Name>
  <Description>
    A tag which contains zero or more CDNames which correspond
    to the CD's that hold type information that this CD depends on. This
    makes an inheritance type structure for CD's. If the CD types are
    dependent on any other CD's they must be present here. For application
    that do not respect types, this symbol can be ignored.
  </Description>
</CDDefinition>

<CDDefinition>
  <Name> Description </Name>
  <Description>
    A tag which contains PCDATA corresponding to the
    description of either the CD or the symbol (depending on which is the
    enclosing element).
  </Description>
</CDDefinition>

<CDDefinition>
  <Name> Name </Name>
  <Description>
    A tag which contains PCDATA corresponding to the name of
    the symbol being defined.
  </Description>
</CDDefinition>

<CDDefinition>
  <Name> Signature </Name>
  <Description>
    An optional tag which contains PCDATA corresponding to
    the type of the symbol being defined. This type information will be
    an &OM; type object as defined in chapter 5 of "First Draft of the
    &OM; Standard".
  </Description>
</CDDefinition>

<CDDefinition>
  <Name> Presentation </Name>
  <Description>
    An optional tag (which may be repeated many times) which contains
    PCDATA corresponding to a way of presenting the symbol being defined.
  </Description>
</CDDefinition>

<CDDefinition>
  <Name> CMP </Name>
  <Description>
    An optional tag (which may be repeated many times) which contains
    PCDATA corresponding to a property of the symbol being
    defined.
  </Description>
</CDDefinition>

<CDDefinition>
  <Name> FMP </Name>
  <Description>
    An optional tag (which may be repeated many times) which contains
    PCDATA corresponding to a property of the symbol being
    defined. This property must be a valid &OM; object.
  </Description>
</CDDefinition>

</CD>
```

A.2 The arith1 Content Dictionary File

```
<CD>
  <CDName> arith1 </CDName>
  <CDURL> http://www.openmath.org/cd/arith1.ocd </CDURL>
  <CDReviewDate> 2003-04-01 </CDReviewDate>
  <CDStatus> official </CDStatus>
  <CDDate> 2001-03-12 </CDDate>
  <CDVersion> 2 </CDVersion>
  <CDRevision> 0 </CDRevision>
  <CDUses>
    <CDName>alg1</CDName>
    <CDName>fns1</CDName>
    <CDName>integer1</CDName>
    <CDName>interval1</CDName>
    <CDName>linalg2</CDName>
    <CDName>logic1</CDName>
    <CDName>quant1</CDName>
    <CDName>relation1</CDName>
    <CDName>set1</CDName>
    <CDName>setname1</CDName>
    <CDName>transc1</CDName>
  </CDUses>

  <Description>
    This CD defines symbols for common arithmetic functions.
  </Description>

  <CDDefinition>
    <Name> lcm </Name>
    <Description>
      The symbol to represent the n-ary function to return the least common
      multiple of its arguments.
    </Description>

    <CMP> lcm(a,b) = a*b/gcd(a,b) </CMP>

    <FMP>
      <OMOBJ>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="lcm"/>
            <OMV name="a"/>
            <OMV name="b"/>
          </OMA>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="divide"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
              <OMV name="a"/>
              <OMV name="b"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="gcd"/>
              <OMV name="a"/>
              <OMV name="b"/>
            </OMA>
          </OMA>
        </OMA>
      </OMOBJ>
    </FMP>
    <CMP>
      for all integers a,b |
      There does not exist a c>0 such that c/a is an Integer and c/b is an
      Integer and lcm(a,b) > c.
    </CMP>

    <FMP>
      <OMOBJ>
        <OMBIND>
```

```xml
              <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
              <OMBVAR>
                <OMV name="a"/> |
                <OMV name="b"/>
              </OMBVAR>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="implies"/>
                <OMA>
                  <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="and"/>
                  <OMA>
                    <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
                    <OMV name="a"/>
                    <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/>
                  </OMA>
                  <OMA>
                    <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
                    <OMV name="b"/>
                    <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/>
                  </OMA>
                </OMA>
                <OMA>
                  <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="not"/>
                  <OMBIND>
                    <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="exists"/>
                    <OMBVAR>
                      <OMV name="c"/>
                    </OMBVAR>
                    <OMA>
                      <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="and"/>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="gt
                        <OMV name="c"/>
                        <OMI>0</OMI>
                      </OMA>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="integer1" name="fac
                        <OMV name="a"/>
                        <OMV name="c"/>
                      </OMA>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="integer1" name="fac
                        <OMV name="b"/>
                        <OMV name="c"/>
                      </OMA>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="lt
                        <OMV name="c"/>
                        <OMA>
                          <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="lc
                          <OMV name="a"/>
                          <OMV name="b"/>
                        </OMA>
                      </OMA>
                    </OMA>
                  </OMBIND>
                </OMA>
              </OMA>
            </OMBIND>
          </OMOBJ>
      </FMP>
   </CDDefinition>

   <CDDefinition>
      <Name> gcd </Name>
      <Description>
        The symbol to represent the n-ary function to return the gcd (greatest
        common divisor) of its arguments.
      </Description>

      <CMP>
        for all integers a,b |
```

```
        There does not exist a c such that a/c is an Integer and b/c is an
        Integer and c > gcd(a,b).

        Note that this implies that gcd(a,b) > 0
</CMP>

<FMP>
    <OMOBJ>
        <OMBIND>
            <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
            <OMBVAR>
                <OMV name="a"/>
                <OMV name="b"/>
            </OMBVAR>
            <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="implies"/>
                <OMA>
                    <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="and"/>
                    <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
                        <OMV name="a"/>
                        <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/>
                    </OMA>
                    <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
                        <OMV name="b"/>
                        <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/>
                    </OMA>
                </OMA>
                <OMA>
                    <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="not"/>
                    <OMBIND>
                        <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="exists"/>
                        <OMBVAR>
                            <OMV name="c"/>
                        </OMBVAR>
                        <OMA>
                            <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="and"/>
                            <OMA>
                                <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
                                <OMA>
                                    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="di
                                    <OMV name="a"/>
                                    <OMV name="c"/>
                                </OMA>
                                <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/
                            </OMA>
                            <OMA>
                                <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
                                <OMA>
                                    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="di
                                    <OMV name="b"/>
                                    <OMV name="c"/>
                                </OMA>
                                <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/
                            </OMA>
                            <OMA>
                                <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="gt
                                <OMV name="c"/>
                                <OMA>
                                    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="gc
                                    <OMV name="a"/>
                                    <OMV name="b"/>
                                </OMA>
                            </OMA>
                        </OMA>
                    </OMBIND>
                </OMA>
            </OMA>
        </OMBIND>
    </OMOBJ>
```

```
      </FMP>

      <Example>
         gcd(6,9) = 3
         <OMOBJ>
           <OMA>
             <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
             <OMA>
               <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="gcd"/>
               <OMI> 6 </OMI>
               <OMI> 9 </OMI>
             </OMA>
             <OMI> 3 </OMI>
           </OMA>
         </OMOBJ>
      </Example>
</CDDefinition>

<CDDefinition>
   <Name> plus </Name>
   <Description>
      The symbol representing an n-ary commutative function plus.
   </Description>
   <CMP> for all a,b | a + b = b + a </CMP>
   <FMP>
      <OMOBJ>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="a"/>
            <OMV name="b"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
            <OMA>
               <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
               <OMV name="a"/>
               <OMV name="b"/>
            </OMA>
            <OMA>
               <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
               <OMV name="b"/>
               <OMV name="a"/>
            </OMA>
          </OMA>
        </OMBIND>
      </OMOBJ>
   </FMP>
</CDDefinition>

<CDDefinition>
   <Name> unary_minus </Name>
   <Description>
      This symbol denotes unary minus, i.e. the additive inverse.
   </Description>
   <CMP> for all a | a + (-a) = 0 </CMP>
   <FMP>
      <OMOBJ>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="a"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
            <OMA>
               <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
               <OMV name="a"/>
               <OMA>
                 <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="unary_min
                 <OMV name="a"/>
```

```
              </OMA>
            </OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="zero"/>
          </OMA>
        </OMBIND>
      </OMOBJ>
    </FMP>
</CDDefinition>

<CDDefinition>
  <Name> minus </Name>
  <Description>
    The symbol representing a binary minus function. This is equivalent to
    adding the additive inverse.
  </Description>
  <CMP> for all a,b | a - b = a + (-b) </CMP>
  <FMP>
      <OMOBJ>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="a"/>
            <OMV name="b"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="minus"/>
              <OMV name="a"/>
              <OMV name="b"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
              <OMV name="a"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="unary_min
                <OMV name="b"/>
              </OMA>
            </OMA>
          </OMA>
        </OMBIND>
      </OMOBJ>
    </FMP>
</CDDefinition>

<CDDefinition>
  <Name> times </Name>
  <Description>
    The symbol representing an n-ary multiplication function.
  </Description>
  <Example>
      <OMOBJ>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrix"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrc
                <OMI> 1 </OMI>
                <OMI> 2 </OMI>
              </OMA>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrc
                <OMI> 3 </OMI>
                <OMI> 4 </OMI>
              </OMA>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrix"/>
```

```
                        <OMA>
                          <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrc
                          <OMI> 5 </OMI>
                          <OMI> 6 </OMI>
                        </OMA>
                        <OMA>
                          <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrc
                          <OMI> 7 </OMI>
                          <OMI> 8 </OMI>
                        </OMA>
                      </OMA>
                    </OMA>
                    <OMA>
                      <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrix"/>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrow"
                        <OMI> 19 </OMI>
                        <OMI> 20 </OMI>
                      </OMA>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrow"
                        <OMI> 43 </OMI>
                        <OMI> 50 </OMI>
                      </OMA>
                    </OMA>
                  </OMA>
                </OMOBJ>
              </Example>
              <CMP> for all a,b | a * 0 = 0 and a * b = a * (b - 1) + a </CMP>

              <FMP><OMOBJ>
                  <OMBIND>
                    <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
                    <OMBVAR>
                      <OMV name="a"/>
                      <OMV name="b"/>
                    </OMBVAR>
                    <OMA>
                      <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="and"/>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
                        <OMA>
                          <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
                          <OMV name="a"/>
                          <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="zero"/>
                        </OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="zero"/>
                      </OMA>
                      <OMA>
                        <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
                        <OMA>
                          <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
                          <OMV name="a"/>
                          <OMV name="b"/>
                        </OMA>
                        <OMA>
                          <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
                          <OMA>
                            <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/
                            <OMV name="a"/>
                            <OMA>
                              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="minus
                              <OMV name="b"/>
                              <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="one"/>
                            </OMA>
                          </OMA>
                          <OMV name="a"/>
                        </OMA>
                      </OMA>
                    </OMA>
                  </OMBIND>
```

```
        </OMOBJ></FMP>

    <CMP> for all a,b,c | a*(b+c) = a*b + a*c </CMP>
    <FMP><OMOBJ>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="a"/>
            <OMV name="b"/>
            <OMV name="c"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
              <OMV name="a"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
                <OMV name="b"/>
                <OMV name="c"/>
              </OMA>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
                <OMV name="a"/>
                <OMV name="b"/>
              </OMA>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
                <OMV name="a"/>
                <OMV name="c"/>
              </OMA>
            </OMA>
          </OMA>
        </OMBIND>
      </OMOBJ></FMP>
</CDDefinition>

<CDDefinition>
  <Name> divide </Name>
  <Description>
    This symbol represents a (binary) division function denoting the first argument
    right-divided by the second, i.e. divide(a,b)=a*inverse(b). It is the
    inverse of the multiplication function defined by the symbol times in this CD.
  </Description>
  <CMP> whenever not(a=0) then a/a = 1 </CMP>
  <FMP>
      <OMOBJ>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="a"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="implies"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="neq"/>
              <OMV name="a"/>
              <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="zero"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="divide"/>
                <OMV name="a"/>
                <OMV name="a"/>
              </OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="one"/>
            </OMA>
```

```
              </OMA>
            </OMBIND>
          </OMOBJ>
      </FMP>
</CDDefinition>

<CDDefinition>
    <Name> power </Name>
    <Description>
        This symbol represents a power function. The first argument is raised
        to the power of the second argument. When the second argument is not
        an integer, powering is defined in terms of exponentials and
        logarithms for the complex and real numbers.
        This operator can represent general powering.
    </Description>

    <CMP>
        x\in C implies x^a = exp(a ln x)
    </CMP>

    <FMP>
        <OMOBJ>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="implies"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
              <OMV name="x"/>
              <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="C"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
              <OMA>
                 <OMS uristub="http://www.openmath.org/cd" name="power" cd="arith1"/>
                 <OMV name="x"/>
                 <OMV name="a"/>
              </OMA>
              <OMA>
                 <OMS uristub="http://www.openmath.org/cd" name="exp" cd="transc1"/>
                 <OMA>
                   <OMS uristub="http://www.openmath.org/cd" name="times" cd="arith1"/>
                   <OMV name="a"/>
                   <OMA>
                     <OMS uristub="http://www.openmath.org/cd" name="ln" cd="transc1"/>
                     <OMV name="x"/>
                   </OMA>
                 </OMA>
              </OMA>
            </OMA>
          </OMA>
        </OMOBJ>
    </FMP>

    <CMP>
        if n is an integer then
        x^0 = 1,
        x^n = x * x^(n-1)
    </CMP>
    <FMP>
        <OMOBJ>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="implies"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
              <OMV name="n"/>
              <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="Z"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="and"/>
              <OMA>
                 <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
                 <OMA>
```

```
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="power"/>
                <OMV name="x"/>
                <OMI>0</OMI>
              </OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="one"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="power"/>
                <OMV name="x"/>
                <OMV name="n"/>
              </OMA>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
                <OMV name="x"/>
                <OMA>
                  <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="power"/
                  <OMV name="x"/>
                  <OMA>
                    <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="minus
                    <OMV name="n"/>
                    <OMI>1</OMI>
                  </OMA>
                </OMA>
              </OMA>
            </OMA>
          </OMA>
        </OMA>
      </OMA>
    </OMOBJ>
</FMP>
<Example>
    <OMOBJ>
      <OMA>
        <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="power"/>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrix"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixro
              <OMI> 1 </OMI>
              <OMI> 2 </OMI>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixro
              <OMI> 3 </OMI>
              <OMI> 4 </OMI>
            </OMA>
          </OMA>
          <OMI>3</OMI>
        </OMA>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrix"/>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrow"
            <OMI> 37 </OMI>
            <OMI> 54 </OMI>
          </OMA>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="linalg2" name="matrixrow"
            <OMI> 81 </OMI>
            <OMI> 118 </OMI>
          </OMA>
        </OMA>
      </OMA>
    </OMOBJ>
</Example>
<Example>
    <OMOBJ>
      <OMA>
```

```xml
            <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="power"/>
              <OMS uristub="http://www.openmath.org/cd" cd="nums1" name="e"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="times"/>
                <OMS uristub="http://www.openmath.org/cd" cd="nums1" name="i"/>
                <OMS uristub="http://www.openmath.org/cd" cd="nums1" name="pi"/>
              </OMA>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="unary_minus"/>
              <OMS uristub="http://www.openmath.org/cd" cd="alg1" name="one"/>
            </OMA>
          </OMA>
        </OMOBJ>
    </Example>
</CDDefinition>

<CDDefinition>
  <Name> abs </Name>
  <Description>
     A unary operator which represents the absolute value of its
     argument. The argument should be numerically valued.
     In the complex case this is often referred to as the modulus.
  </Description>
  <CMP> for all x,y | abs(x) + abs(y) >= abs(x+y) </CMP>
  <FMP>
     <OMOBJ>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
          <OMBVAR>
            <OMV name="x"/>
            <OMV name="y"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="geq"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="abs"/>
                <OMV name="x"/>
                <OMV name="y"/>
              </OMA>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="abs"/>
                <OMV name="x"/>
                <OMV name="y"/>
              </OMA>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="abs"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plus"/>
                <OMV name="x"/>
                <OMV name="y"/>
              </OMA>
            </OMA>
          </OMA>
        </OMBIND>
     </OMOBJ>
  </FMP>
</CDDefinition>

<CDDefinition>
  <Name> root </Name>
  <Description>
     A binary operator which represents its first argument "lowered" to its
     n'th root where n is the second argument. This is the inverse of the operation
     represented by the power symbol defined in this CD.
```

```
        Care should be taken as to the precise meaning of this operator, in
        particular which root is represented, however it is here to represent
        the general notion of taking n'th roots. As inferred by the signature
        relevant to this symbol, the function represented by this symbol is
        the single valued function, the specific root returned is the one
        indicated by the first CMP. Note also that the converse of the second
        CMP is not valid in general.
      </Description>

      <CMP> x\in C implies root(x,n) = exp(ln(x)/n) </CMP>
      <FMP>
        <OMOBJ>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="logic1" name="implies"/>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="set1" name="in"/>
              <OMV name="x"/>
              <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="C"/>
            </OMA>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="root"/>
                <OMV name="x"/>
                <OMV name="n"/>
              </OMA>
              <OMA>
                <OMS name="exp" cd="transc1"/>
                <OMA>
                  <OMS name="divide" cd="arith1"/>
                  <OMA>
                    <OMS name="ln" cd="transc1"/>
                    <OMV name="x"/>
                  </OMA>
                  <OMV name="n"/>
                </OMA>
              </OMA>
            </OMA>
          </OMA>
        </OMOBJ>
      </FMP>

      <CMP> for all a,n | power(root(a,n),n) = a (if the root exists!) </CMP>
      <FMP>
        <OMOBJ>
          <OMBIND>
            <OMS uristub="http://www.openmath.org/cd" cd="quant1" name="forall"/>
            <OMBVAR>
              <OMV name="a"/>
              <OMV name="n"/>
            </OMBVAR>
            <OMA>
              <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
              <OMA>
                <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="power"/>
                <OMA>
                  <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="root"/>
                  <OMV name="a"/>
                  <OMV name="n"/>
                </OMA>
                <OMV name="n"/>
              </OMA>
              <OMV name="a"/>
            </OMA>
          </OMBIND>
        </OMOBJ>
      </FMP>
    </CDDefinition>
```

```
<CDDefinition>
  <Name> sum </Name>
  <Description>
    An operator taking two arguments, the first being the range of summation,
    e.g. an integral interval, the second being the function to be
    summed. Note that the sum may be over an infinite interval.
  </Description>
  <Example>
    This represents the summation of the reciprocals of all the integers between
    1 and 10 inclusive.
    <OMOBJ>
      <OMA>
        <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="sum"/>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="interval1" name="integer_int
          <OMI> 1 </OMI>
          <OMI> 10 </OMI>
        </OMA>
        <OMBIND>
          <OMS uristub="http://www.openmath.org/cd" cd="fns1" name="lambda"/>
          <OMBVAR>
            <OMV name="x"/>
          </OMBVAR>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="divide"/>
            <OMI> 1 </OMI>
            <OMV name="x"/>
          </OMA>
        </OMBIND>
      </OMA>
    </OMOBJ>
  </Example>
</CDDefinition>

<CDDefinition>
  <Name> product </Name>
  <Description>
    An operator taking two arguments, the first being the range of multiplication
    e.g. an integral interval, the second being the function to
    be multiplied. Note that the product may be over an infinite interval.
  </Description>
  <Example>
    This represents the statement that the factorial of n is equal to the product
    of all the integers between 1 and n inclusive.
    <OMOBJ>
      <OMA>
        <OMS uristub="http://www.openmath.org/cd" cd="relation1" name="eq"/>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="integer1" name="factorial"/>
          <OMV name="n" />
        </OMA>
        <OMA>
          <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="product"/>
          <OMA>
            <OMS uristub="http://www.openmath.org/cd" cd="interval1" name="integer_
            <OMI> 1 </OMI>
            <OMV name="n"/>
          </OMA>
          <OMBIND>
            <OMS uristub="http://www.openmath.org/cd" cd="fns1" name="lambda"/>
            <OMBVAR>
              <OMV name="i"/>
            </OMBVAR>
            <OMV name="i"/>
          </OMBIND>
        </OMA>
      </OMA>
    </OMOBJ>
  </Example>
</CDDefinition>
```

```
    </CD>
```

A.3 The arith1 STS Signature File

```
<CDSignatures type="sts" cd="arith1">

  <CDSComment>
    Date:  1999-11-26
    Author: David Carlisle
  </CDSComment>

  <Signature name="lcm" >
    <OMOBJ>
       <OMA>
         <OMS name="mapsto" cd="sts" />
         <OMA>
           <OMS name="nassoc" cd="sts"/>
           <OMV name="SemiGroup"/>
         </OMA>
         <OMV name="SemiGroup" />
       </OMA>
    </OMOBJ>
  </Signature>

  <Signature name="gcd" >
    <OMOBJ>
       <OMA>
         <OMS name="mapsto" cd="sts" />
         <OMA>
           <OMS name="nassoc" cd="sts"/>
           <OMV name="SemiGroup"/>
         </OMA>
         <OMV name="SemiGroup" />
       </OMA>
    </OMOBJ>
  </Signature>

  <Signature name="plus">
    <OMOBJ>
       <OMA>
         <OMS name="mapsto" cd="sts"/>
         <OMA>
           <OMS name="nassoc" cd="sts"/>
           <OMV name="AbelianSemiGroup"/>
         </OMA>
         <OMV name="AbelianSemiGroup"/>
       </OMA>
    </OMOBJ>
  </Signature>

  <Signature name="unary_minus">
    <OMOBJ>
       <OMA>
         <OMS name="mapsto" cd="sts"/>
         <OMV name="AbelianGroup"/>
         <OMV name="AbelianGroup"/>
       </OMA>
    </OMOBJ>
  </Signature>

  <Signature name="minus">
    <OMOBJ>
       <OMA>
         <OMS name="mapsto" cd="sts"/>
```

```xml
        <OMV name="AbelianGroup"/>
        <OMV name="AbelianGroup"/>
        <OMV name="AbelianGroup"/>
      </OMA>
    </OMOBJ>
</Signature>

<Signature name="times">
  <OMOBJ>
      <OMA>
        <OMS name="mapsto" cd="sts"/>
        <OMA>
          <OMS name="nassoc" cd="sts"/>
          <OMV name="SemiGroup"/>
        </OMA>
        <OMV name="SemiGroup"/>
      </OMA>
  </OMOBJ>
</Signature>

<Signature name="divide">
  <OMOBJ>
      <OMA>
        <OMS name="mapsto" cd="sts"/>
        <OMV name="AbelianGroup"/>
        <OMV name="AbelianGroup"/>
        <OMV name="AbelianGroup"/>
      </OMA>
  </OMOBJ>
</Signature>

<Signature name="power">
  <OMOBJ>
      <OMA>
        <OMS name="mapsto" cd="sts"/>
        <OMS name="NumericalValue" cd="sts"/>
        <OMS name="NumericalValue" cd="sts"/>
        <OMS name="NumericalValue" cd="sts"/>
      </OMA>
  </OMOBJ>
</Signature>

<Signature name="abs">
  <OMOBJ>
      <OMA>
        <OMS name="mapsto" cd="sts"/>
        <OMS name="C" cd="setname1"/>
        <OMS name="R" cd="setname1"/>
      </OMA>
  </OMOBJ>
</Signature>

<Signature name="root">
  <OMOBJ>
      <OMA>
        <OMS name="mapsto" cd="sts"/>
        <OMS name="NumericalValue" cd="sts"/>
        <OMS name="NumericalValue" cd="sts"/>
        <OMS name="NumericalValue" cd="sts"/>
      </OMA>
  </OMOBJ>
</Signature>


<Signature name="sum" >
  <OMOBJ>
      <OMA>
        <OMS name="mapsto" cd="sts" />
        <OMV name="IntegerRange" />
        <OMA>
          <OMS name="mapsto" cd="sts" />
```

```
        <OMS name="Z" cd="setname1" />
        <OMV name="AbelianMonoid" />
      </OMA>
      <OMV name="AbelianMonoid" />
    </OMA>
  </OMOBJ>
</Signature>

<Signature name="product" >
  <OMOBJ>
    <OMA>
      <OMS name="mapsto" cd="sts" />
      <OMV name="IntegerRange" />
      <OMA>
        <OMS name="mapsto" cd="sts" />
        <OMS name="Z" cd="setname1" />
        <OMV name="AbelianMonoid" />
      </OMA>
      <OMV name="AbelianMonoid" />
    </OMA>
  </OMOBJ>
</Signature>

</CDSignatures>
```

A.4 The MathML CDGroup

```
<CDGroup>
  <CDGroupName>mathml</CDGroupName>
  <CDGroupVersion> 2 </CDGroupVersion>
  <CDGroupRevision> 0 </CDGroupRevision>
  <CDGroupURL>
    http://www.openmath.org/cdfiles/cdgroups/mathml.ocd</CDGroupURL>
  <CDGroupDescription> MathML compatibility CD Group </CDGroupDescription>
  <CDComment>This is the first version of the Core CD group.
    It was created by D Carlisle based on MathML CD Group.</CDComment>
  <CDComment>Algebra</CDComment>
  <CDGroupMember>
    <CDName>alg1</CDName>
    <CDURL>http://www.openmath.org/cd/alg1.ocd</CDURL></CDGroupMember>
  <CDComment>Arithmetic</CDComment>
  <CDGroupMember>
    <CDName>arith1</CDName>
    <CDURL>http://www.openmath.org/cd/arith1.ocd</CDURL></CDGroupMember>
  <CDComment>Constructor for Floating Point Numbers</CDComment>
  <CDGroupMember>
    <CDName>bigfloat1</CDName>
    <CDURL>http://www.openmath.org/cd/bigfloat1.ocd</CDURL></CDGroupMember>
  <CDComment>Calculus</CDComment>
  <CDGroupMember>
    <CDName>calculus1</CDName>
    <CDURL>http://www.openmath.org/cd/calculus1.ocd</CDURL></CDGroupMember>
  <CDComment>Operations on and constructors for complex numbers</CDComment>
  <CDGroupMember>
    <CDName>complex1</CDName>
    <CDURL>http://www.openmath.org/cd/complex1.ocd</CDURL></CDGroupMember>
  <CDComment>Functions on functions</CDComment>
  <CDGroupMember>
    <CDName>fns1</CDName>
    <CDURL>http://www.openmath.org/cd/fns1.ocd</CDURL></CDGroupMember>
  <CDComment>Integer arithmetic</CDComment>
  <CDGroupMember>
    <CDName>integer1</CDName>
    <CDURL>http://www.openmath.org/cd/integer1.ocd</CDURL></CDGroupMember>
  <CDComment>Intervals</CDComment>
```

```
<CDGroupMember>
  <CDName>interval1</CDName>
  <CDURL>http://www.openmath.org/cd/interval1.ocd</CDURL></CDGroupMember>
<CDComment>Linear Algebra - vector &amp; matrix constructors, those symbols which are
<CDGroupMember>
  <CDName>linalg1</CDName>
  <CDURL>http://www.openmath.org/cd/linalg1.ocd</CDURL></CDGroupMember>
<CDComment>Linear Algebra - vector &amp; matrix constructors, those symbols which are
<CDGroupMember>
  <CDName>linalg2</CDName>
  <CDURL>http://www.openmath.org/cd/linalg2.ocd</CDURL></CDGroupMember>
<CDComment>Limits of unary functions</CDComment>
<CDGroupMember>
  <CDName>limit1</CDName>
  <CDURL>http://www.openmath.org/cd/limit1.ocd</CDURL></CDGroupMember>
<CDComment>List constructors</CDComment>
<CDGroupMember>
  <CDName>list1</CDName>
  <CDURL>http://www.openmath.org/cd/list1.ocd</CDURL></CDGroupMember>
<CDComment>Basic logical operators</CDComment>
<CDGroupMember>
  <CDName>logic1</CDName>
  <CDURL>http://www.openmath.org/cd/logic1.ocd</CDURL></CDGroupMember>
<CDComment>
  MathML Numerical Types
</CDComment>
<CDGroupMember>
  <CDName>mathmltypes</CDName>
  <CDURL>http://www.openmath.org/cd/mathmltypes.ocd</CDURL>
</CDGroupMember>
<CDComment>Minima and maxima</CDComment>
<CDGroupMember>
  <CDName>minmax1</CDName>
  <CDURL>http://www.openmath.org/cd/minmax1.ocd</CDURL></CDGroupMember>
<CDComment>Multset-theoretic operators and constructors</CDComment>
<CDGroupMember>
  <CDName>multiset1</CDName>
  <CDURL>http://www.openmath.org/cd/multiset1.ocd</CDURL></CDGroupMember>
<CDComment>Symbols for creating numbers, including some defined constants
  (which can be seen as nullary constructors)</CDComment>
<CDGroupMember>
  <CDName>nums1</CDName>
  <CDURL>http://www.openmath.org/cd/nums1.ocd</CDURL></CDGroupMember>
<CDComment>Symbols for creating piecewise definitions</CDComment>
<CDGroupMember>
  <CDName>piece1</CDName>
  <CDURL>http://www.openmath.org/cd/piece1.ocd</CDURL></CDGroupMember>
<CDComment>The basic quantifiers forall and exists.</CDComment>
<CDGroupMember>
  <CDName>quant1</CDName>
  <CDURL>http://www.openmath.org/cd/quant1.ocd</CDURL></CDGroupMember>
<CDComment>Common arithmetic relations</CDComment>
<CDGroupMember>
  <CDName>relation1</CDName>
  <CDURL>http://www.openmath.org/cd/relation1.ocd</CDURL></CDGroupMember>
<CDComment>Number sets</CDComment>
<CDGroupMember>
  <CDName>setname1</CDName>
  <CDURL>http://www.openmath.org/cd/setname1.ocd</CDURL></CDGroupMember>
<CDComment>Rounding</CDComment>
<CDGroupMember>
  <CDName>rounding1</CDName>
  <CDURL>http://www.openmath.org/cd/rounding1.ocd</CDURL></CDGroupMember>
<CDComment>Set-theoretic operators and constructors</CDComment>
<CDGroupMember>
  <CDName>set1</CDName>
  <CDURL>http://www.openmath.org/cd/set1.ocd</CDURL></CDGroupMember>
<CDComment>Basic data orientated statistical operators</CDComment>
<CDGroupMember>
  <CDName>s_data1</CDName>
  <CDURL>http://www.openmath.org/cd/s_data1.ocd</CDURL></CDGroupMember>
```

```
      <CDComment>Basic random variable orientated statistical operators</CDComment>
      <CDGroupMember>
        <CDName>s_dist1</CDName>
        <CDURL>http://www.openmath.org/cd/s_dist1.ocd</CDURL></CDGroupMember>
      <CDComment>Basic transcendental functions</CDComment>
      <CDGroupMember>
        <CDName>transc1</CDName>
        <CDURL>http://www.openmath.org/cd/transc1.ocd</CDURL></CDGroupMember>
      <CDComment>Vector calculus functions</CDComment>
      <CDGroupMember>
        <CDName>veccalc1</CDName>
        <CDURL>http://www.openmath.org/cd/veccalc1.ocd</CDURL></CDGroupMember>
      <CDComment>Alternative encoding symbols for compatibility with the MathML
        Semantic mapping constructs.</CDComment>
      <CDGroupMember>
        <CDName>altenc</CDName>
        <CDURL>http://www.openmath.org/cd/altenc.ocd</CDURL></CDGroupMember>
    </CDGroup>
```

A.5 The error Content Dictionary

```
<CD>
  <CDName> error </CDName>
  <CDURL> http://www.openmath.org/cd/error.ocd </CDURL>
  <CDReviewDate> 2003-04-01 </CDReviewDate>
  <CDStatus> official </CDStatus>
  <CDDate> 2001-03-12 </CDDate>
  <CDVersion> 2 </CDVersion>
  <CDRevision> 0 </CDRevision>
  <CDUses>
    <CDName> arith1 </CDName>
    <CDName> specfun1 </CDName>
  </CDUses>

  <CDDefinition>
    <Name> unhandled_symbol </Name>
    <Description>
      This symbol represents the error which is raised when an application
      reads a symbol which is present in the mentioned content
      dictionary, but which it has not implemented.

      When receiving such a symbol, the application should act as if it had
      received the &OM; error object constructed from unhandled_symbol
      and the unhandled symbol as in the example below.
    </Description>

    <Example>
      The application does not implement the Complex numbers:
      <OMOBJ>
        <OME>
          <OMS uristub="http://www.openmath.org/cd" cd="error" name="unhandled_symbol"
          <OMS uristub="http://www.openmath.org/cd" cd="setname1" name="C"/>
        </OME>
      </OMOBJ>
    </Example>
  </CDDefinition>

  <CDDefinition>
    <Name> unexpected_symbol </Name>
    <Description>
      This symbol represents the error which is raised when an application
      reads a symbol which is not present in the mentioned content dictionary.

      When receiving such a symbol, the application should act as if it had
      received the &OM; error object constructed from unexpected_symbol
```

```
            and the unexpected symbol as in the example below.
        </Description>
        <Example>
           The application received a mistyped symbol
           <OMOBJ>
             <OME>
               <OMS uristub="http://www.openmath.org/cd" cd="error" name="unexpected_symbol
               <OMS uristub="http://www.openmath.org/cd" cd="arith1" name="plurse"/>
             </OME>
           </OMOBJ>
        </Example>
     </CDDefinition>

     <CDDefinition>
        <Name> unsupported_CD </Name>
        <Description>
           This symbol represents the error which is raised when an application
           reads a symbol where the mentioned content dictionary is not
           present.

           When receiving such a symbol, the application should act as if it had
           received the &OM; error object constructed from unsupported_CD and
           the symbol from the unsupported Content Dictionary as in the example
           below.
        </Description>
        <Example>
           The application does not know about the CD specfun1
           <OMOBJ>
             <OME>
               <OMS uristub="http://www.openmath.org/cd" cd="error" name="unsupported_CD"/>
               <OMS uristub="http://www.openmath.org/cd" cd="specfun1" name="BesselJ"/>
             </OME>
           </OMOBJ>
        </Example>
     </CDDefinition>

  </CD>
```

Appendix B
OpenMath Schema in Relax NG Compact Syntax (Non-Normative)

This is the Relax NG Schema described in expressed according to the Relax NG Compact Syntax described in [11].

```
    # Schema for OpenMath 2



default namespace om = "http://www.openmath.org/OpenMath"

namespace xlink = "http://www.w3.org/1999/xlink"



# OM2: allow OMR

omel =

  OMS | OMV | OMI | OMB | OMSTR | OMF | OMA | OMBIND | OME | OMATTR |OMR
```

```
# things which can be variables

omvar = OMV | attvar


attvar = element OMATTR { common.attributes,(OMATP , (OMV| attvar))}


#OM2: common.attributes attributes

common.attributes =

  (attribute id { xsd:ID })?


# symbol

OMS = element OMS { common.attributes, attlist.OMS}

attlist.OMS =

  attribute name { xsd:NCName},

  attribute cd { xsd:NCName},

  attribute uristub { xsd:anyURI}


# variable

OMV = element OMV { common.attributes, attlist.OMV}

attlist.OMV = attribute name { xsd:NCName}


# integer

OMI = element OMI { common.attributes, xsd:string {pattern = "\s*(-\s?)?[0-9]+(\s[0-9]+)'

# byte array

OMB = element OMB { common.attributes, xsd:base64Binary }


# string

OMSTR = element OMSTR { common.attributes, text }


# floating point

OMF = element OMF { common.attributes, attlist.OMF}

attlist.OMF =

  attribute dec { xsd:string {pattern = "(-?)([0-9]+)?(\.[0-9]+)?(e([+\-]?)[0-9]+)?"}}|

  attribute hex { xsd:string {pattern = "[0-9A-F]+"}}
```

```
# apply constructor

OMA = element OMA { common.attributes, omel+ }

# binding constructor and variable

OMBIND = element OMBIND { common.attributes, omel, OMBVAR, omel }

OMBVAR = element OMBVAR { common.attributes, omvar+ }


# error

OME = element OME { common.attributes, OMS, omel* }

# attribution constructor and attribute pair constructor

OMATTR = element OMATTR { common.attributes, OMATP, omel }


# OM2: allow OMFOREIGN

OMATP = element OMATP { common.attributes, (OMS, (omel | OMFOREIGN) )+ }


# OM2: OMFOREIGN

OMFOREIGN =  element OMFOREIGN { common.attributes, (omel|notom)* }


# Any elements not in the om namespace (valid om is allowed as a descendant)

notom =

  (element * - om:* {

    attribute * { text }*,

    (omel|notom)*

  }

  | text)




# OM object constructor

OMOBJ = element OMOBJ { common.attributes, omel }

 attlist.OMOBJ = attribute version { "1.0" | "1.2" | "2.0" }

start = OMOBJ


# OM2: OMR

OMR = element OMR { common.attributes, attlist.OMR }

attlist.OMR =

        attribute xlink:href { text },
```

```
            attribute xlink:type {"simple"},

            attribute xlink:show {"embed"}
```

Appendix C
Restricting the *OpenMath* Schema (Non-Normative)

Relax NG allows one to state constraints such as *if the cd attribute of OMS is arith1 then the name attribute must be one of lcm, gcd, plus etc.* Thus it is easy to use a stylesheet to generate for any given CD, a Relax NG schema that expresses the constraint that an OMS naming that CD must only use symbols defined in the specified dictionary. Similarly it is possible to use the *role* information contained in the CD to restrict which symbols can be the first child of an OMBIND or the odd - numbered children of an OMATP.

The modularisation mechanisms of Relax NG then allow one to include these schema for all the CDs that you want to allow and, for example, to replace the regexp -based validation of the OMS attributes by explicit lists of allowed CD names, and for each CD Name, a list of allowed symbol names.

For example, a CD-specific Relax NG Schema for the arith1 CD shown in [Appendix A.2](#) would look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0" datatypeLibrary="">
  <define name="cd.attlist.OMS" combine="choice">
    <attribute name="cd">
      <value type="string">arith1</value>
    </attribute>
    <attribute name="name">
      <choice>
        <value type="string">lcm</value>
        <value type="string">gcd</value>
        <value type="string">plus</value>
        <value type="string">unary_minus</value>
        <value type="string">minus</value>
        <value type="string">times</value>
        <value type="string">divide</value>
        <value type="string">power</value>
        <value type="string">abs</value>
        <value type="string">root</value>
        <value type="string">sum</value>
        <value type="string">product</value>
      </choice>
    </attribute>
  </define>
</grammar>
```

or, using the Relax NG compact syntax:

```
cd.attlist.OMS |=
  attribute cd {string "arith1" },
  attribute name {
```

```
        string "lcm" |
        string "gcd" |
        string "plus" |
        string "unary_minus" |
        string "minus" |
        string "times" |
        string "divide" |
        string "power" |
        string "abs" |
        string "root" |
        string "sum" |
        string "product" }
```

To build a schema that allows only symbols from arith1 we just need to include the *OpenMath* schema described in Appendix B, override the attribute declarations for OMS, and then include the schema for arith1. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
  <include href="openmath.rng">
    <define name="attlist.OMS">
      <ref name="cd.attlist.OMS"/>
    </define>
  </include>
  <include href="arith1.rng"/>
</grammar>
```

or, in the compact syntax:

```
include "openmath.rnc" {
 attlist.OMS = cd.attlist.OMS}

include "arith1.rnc"
```

Using this approach it is possible to include as many files as required.

Appendix D
OpenMath Schema in XSD Syntax (Non-Normative)

This is an XSD Schema generated from the Relax NG Schema described in Section 4.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Schema for OpenMath 2 -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
           targetNamespace="http://www.openmath.org/OpenMath"
           xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:om="http://www.openmath.org/OpenMath">

  <xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
  <!-- OM2: allow OMR -->
  <xs:group name="omel">
    <xs:choice>
      <xs:element ref="om:OMS"/>
      <xs:element ref="om:OMV"/>
```

```xml
      <xs:element ref="om:OMI"/>
      <xs:element ref="om:OMB"/>
      <xs:element ref="om:OMSTR"/>
      <xs:element ref="om:OMF"/>
      <xs:element ref="om:OMA"/>
      <xs:element ref="om:OMBIND"/>
      <xs:element ref="om:OME"/>
      <xs:group ref="om:OMATTR"/>
      <xs:element ref="om:OMR"/>
  </xs:choice>
</xs:group>
<!-- things which can be variables -->
<xs:group name="omvar">
  <xs:choice>
    <xs:element ref="om:OMV"/>
    <xs:group ref="om:attvar"/>
  </xs:choice>
</xs:group>
<xs:group name="attvar">
  <xs:sequence>
    <xs:element name="OMATTR">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="om:OMATP"/>
          <xs:choice>
            <xs:element ref="om:OMV"/>
            <xs:group ref="om:attvar"/>
          </xs:choice>
        </xs:sequence>
        <xs:attributeGroup ref="om:common.attributes"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:group>
<!-- OM2: common.attributes attributes -->
<xs:attributeGroup name="common.attributes">
  <xs:attribute name="id" type="xs:ID"/>
</xs:attributeGroup>
<!-- symbol -->
<xs:element name="OMS">
  <xs:complexType>
    <xs:attributeGroup ref="om:common.attributes"/>
    <xs:attributeGroup ref="om:attlist.OMS"/>
  </xs:complexType>
</xs:element>
<xs:attributeGroup name="attlist.OMS">
  <xs:attribute name="name" use="required" type="xs:NCName"/>
  <xs:attribute name="cd" use="required" type="xs:NCName"/>
  <xs:attribute name="uristub" use="required" type="xs:anyURI"/>
</xs:attributeGroup>
<!-- variable -->
<xs:element name="OMV">
  <xs:complexType>
    <xs:attributeGroup ref="om:common.attributes"/>
    <xs:attributeGroup ref="om:attlist.OMV"/>
  </xs:complexType>
</xs:element>
<xs:attributeGroup name="attlist.OMV">
  <xs:attribute name="name" use="required" type="xs:NCName"/>
</xs:attributeGroup>
<!-- integer -->
<xs:element name="OMI">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="xs:anyType">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="\s*(-\s?)?[0-9]+(\s[0-9]+)*\s*"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:attributeGroup ref="om:common.attributes"/>
```

```xml
            </xs:restriction>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <!-- byte array -->
    <xs:element name="OMB">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:base64Binary">
            <xs:attributeGroup ref="om:common.attributes"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <!-- string -->
    <xs:element name="OMSTR">
      <xs:complexType mixed="true">
        <xs:attributeGroup ref="om:common.attributes"/>
      </xs:complexType>
    </xs:element>
    <!-- floating point -->
    <xs:element name="OMF">
      <xs:complexType>
        <xs:attributeGroup ref="om:common.attributes"/>
        <xs:attributeGroup ref="om:attlist.OMF"/>
      </xs:complexType>
    </xs:element>
    <xs:attributeGroup name="attlist.OMF">
      <xs:attribute name="dec">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="(-?)([0-9]+)?(\.[0-9]+)?(e([+\-]?)[0-9]+)?"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="hex">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:pattern value="[0-9A-F]+"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:attributeGroup>
    <!-- apply constructor -->
    <xs:element name="OMA">
      <xs:complexType>
        <xs:group maxOccurs="unbounded" ref="om:omel"/>
        <xs:attributeGroup ref="om:common.attributes"/>
      </xs:complexType>
    </xs:element>
    <!-- binding constructor and variable -->
    <xs:element name="OMBIND">
      <xs:complexType>
        <xs:sequence>
          <xs:group ref="om:omel"/>
          <xs:element ref="om:OMBVAR"/>
          <xs:group ref="om:omel"/>
        </xs:sequence>
        <xs:attributeGroup ref="om:common.attributes"/>
      </xs:complexType>
    </xs:element>
    <xs:element name="OMBVAR">
      <xs:complexType>
        <xs:group maxOccurs="unbounded" ref="om:omvar"/>
        <xs:attributeGroup ref="om:common.attributes"/>
      </xs:complexType>
    </xs:element>
    <!-- error -->
    <xs:element name="OME">
      <xs:complexType>
        <xs:sequence>
```

```xml
        <xs:element ref="om:OMS"/>
        <xs:group minOccurs="0" maxOccurs="unbounded" ref="om:omel"/>
      </xs:sequence>
      <xs:attributeGroup ref="om:common.attributes"/>
    </xs:complexType>
  </xs:element>
  <!-- attribution constructor and attribute pair constructor -->
  <xs:group name="OMATTR">
    <xs:sequence>
      <xs:element name="OMATTR">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="om:OMATP"/>
            <xs:group ref="om:omel"/>
          </xs:sequence>
          <xs:attributeGroup ref="om:common.attributes"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>
  <!-- OM2: allow OMFOREIGN -->
  <xs:element name="OMATP">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element ref="om:OMS"/>
        <xs:choice>
          <xs:group ref="om:omel"/>
          <xs:element ref="om:OMFOREIGN"/>
        </xs:choice>
      </xs:sequence>
      <xs:attributeGroup ref="om:common.attributes"/>
    </xs:complexType>
  </xs:element>
  <!-- OM2: OMFOREIGN -->
  <xs:element name="OMFOREIGN">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:group ref="om:omel"/>
        <xs:group ref="om:notom"/>
      </xs:choice>
      <xs:attributeGroup ref="om:common.attributes"/>
    </xs:complexType>
  </xs:element>
  <!-- Any elements not in the om namespace (valid om is allowed as a descendant) -->
  <xs:group name="notom">
    <xs:sequence>
      <xs:choice minOccurs="0">
        <xs:any namespace="##other" processContents="skip"/>
        <xs:any namespace="##local" processContents="skip"/>
      </xs:choice>
    </xs:sequence>
  </xs:group>
  <!-- OM object constructor -->
  <xs:element name="OMOBJ">
    <xs:complexType>
      <xs:group ref="om:omel"/>
      <xs:attributeGroup ref="om:common.attributes"/>
    </xs:complexType>
  </xs:element>
  <xs:attributeGroup name="attlist.OMOBJ">
    <xs:attribute name="version" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="1.0"/>
          <xs:enumeration value="1.2"/>
          <xs:enumeration value="2.0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:attributeGroup>
  <!-- OM2: OMR -->
```

```
  <xs:element name="OMR">
    <xs:complexType>
      <xs:attributeGroup ref="om:common.attributes"/>
      <xs:attributeGroup ref="om:attlist.OMR"/>
    </xs:complexType>
  </xs:element>
  <xs:attributeGroup name="attlist.OMR">
    <xs:attribute ref="xlink:href" use="required"/>
    <xs:attribute ref="xlink:type" use="required"/>
    <xs:attribute ref="xlink:show" use="required"/>
  </xs:attributeGroup>
</xs:schema>
```

Appendix E
OpenMath DTD (Non-Normative)

This is a DTD generated from the Relax NG Schema described in   Section 4.1. Note that we cannot express the fact that the   OMFOREIGN element can contain any well -formed XML, so we have simply restricted it to contain any XML defined in the DTD.

```
<?xml encoding="UTF-8"?>

<!-- DTD for OpenMath 2 -->

<!-- OM2: allow OMR -->

<!ENTITY % omel "OMS|OMV|OMI|OMB|OMSTR|OMF|OMA|OMBIND|OME|OMATTR|OMR">

<!ENTITY % attvar "OMATTR">

<!-- things which can be variables -->

<!ENTITY % omvar "OMV|%attvar;">

<!-- OM2: common.attributes attributes -->

<!ENTITY % common.attributes "
  id ID #IMPLIED">

<!ENTITY % attlist.OMS "
  name NMTOKEN #REQUIRED
  cd NMTOKEN #REQUIRED
  uristub CDATA #REQUIRED">

<!-- symbol -->

<!ELEMENT OMS EMPTY>
<!ATTLIST OMS
  %common.attributes;
  %attlist.OMS;>

<!ENTITY % attlist.OMV "
  name NMTOKEN #REQUIRED">

<!-- variable -->

<!ELEMENT OMV EMPTY>
<!ATTLIST OMV
  %common.attributes;
```

```
    %attlist.OMV;>

<!-- integer -->

<!ELEMENT OMI (#PCDATA)>
<!ATTLIST OMI
   %common.attributes;>

<!-- byte array -->

<!ELEMENT OMB (#PCDATA)>
<!ATTLIST OMB
   %common.attributes;>

<!-- string -->

<!ELEMENT OMSTR (#PCDATA)>
<!ATTLIST OMSTR
   %common.attributes;>

<!ENTITY % attlist.OMF "
   dec CDATA #IMPLIED
   hex CDATA #IMPLIED">

<!-- floating point -->

<!ELEMENT OMF EMPTY>
<!ATTLIST OMF
   %common.attributes;
   %attlist.OMF;>

<!-- apply constructor -->

<!ELEMENT OMA (%omel;)+>
<!ATTLIST OMA
   %common.attributes;>

<!-- binding constructor and variable -->

<!ELEMENT OMBIND ((%omel;),OMBVAR,(%omel;))>
<!ATTLIST OMBIND
   %common.attributes;>

<!ELEMENT OMBVAR (%omvar;)+>
<!ATTLIST OMBVAR
   %common.attributes;>

<!-- error -->

<!ELEMENT OME (OMS,(%omel;)*)>
<!ATTLIST OME
   %common.attributes;>

<!-- attribution constructor and attribute pair constructor -->

<!ELEMENT OMATTR (OMATP,(%omel;))>
<!ATTLIST OMATTR
   %common.attributes;>

<!-- OM2: allow OMFOREIGN  -->

<!ELEMENT OMATP (OMS,(%omel;|OMFOREIGN))+>
<!ATTLIST OMATP
   %common.attributes;>

<!-- OM2: OMFOREIGN - note that for this DTD the content is restricted
                    to OpenMath -->

<!ELEMENT OMFOREIGN (%omel;)*>
<!ATTLIST OMFOREIGN
   %common.attributes;>
```

```
<!-- OM object constructor -->

<!ELEMENT OMOBJ (%omel;)>
<!ATTLIST OMOBJ
  %common.attributes;>

<!ENTITY % attlist.OMOBJ "
  version (1.0|1.2|2.0) #REQUIRED">

<!ENTITY % attlist.OMR "
  xlink:href CDATA #REQUIRED
  xlink:type (simple) #REQUIRED
  xlink:show (embed) #REQUIRED">

<!-- OM2: OMR -->

<!ELEMENT OMR EMPTY>
<!ATTLIST OMR
  xmlns:xlink CDATA #FIXED 'http://www.w3.org/1999/xlink'
  %common.attributes;
  %attlist.OMR;>
<!ATTLIST OMOBJ xmlns CDATA  #FIXED "http://www.openmath.org/OpenMath">
```

Appendix F
Changes between *OpenMath* 1.1 and *OpenMath* 2 (Non-Normative)

In this appendix we describe the major changes that occurred between version 1.1 and version 2 of the *OpenMath* standard. All changes to the encodings and content dictionaries have been designed to be backward compatibile, in other words all existing *OpenMath* objects and Content Dictionaries are still valid in *OpenMath* 2. On the other hand an existing *OpenMath* 1.1 application may not be able to process *OpenMath* 2 objects.

F.1 Changes to the Formal Definition of Objects

Additional features of abstract objects have been introduced:

- *OpenMath* symbols have an optional r ôle qualifier which restricts the place where they may occur within compound objects. Although part of the abstract description of a symbol this information is intended to be stored in the CD. In the xml encoding it may be used to provide a more restricted schema leading to tighter validation.

- In addition to their *name* and *cd* properties, symbols now have an optional *uristub* property. This can be used to disambiguate between two CDs which are produced independently but have the same name, and can also be used to produce a canonical URI for any *OpenMath* symbol for use in frameworks such as RDFS or MathML which need one.

- *OpenMath* variables can be indexed, in which case they carry information about their enumerator. Since *OpenMath* only allows variables and not objects to be bound, it was not possible to produce indexed variables by defining an appropriate symbol in a CD. Also, although it would be possible to produce an indexed variable using the new *semantic-attribute* role, it was felt that attributes describe or modify an object, whereas in this case we were constructing an atomic object.

- An *OpenMath* object may be attributed with a non -*OpenMath* object using the new *foreign* constructor. This allows an xml-encoded *OpenMath* object to be attributed with appropriate Presentation MathML, for example, or a base -64 encoded MPEG file of its aural rendering.

- the new role property can be used to indicate that a symbol is an *attribution*, in which case an application may ignore or remove it, or a *semantic attribution* in which case removing it is no longer guarenteed to produce an equivalent object.

- restrictions on the names of symbols, variables and content dictionaries have been relaxed to be compatible with XML and to be less Anglo-Saxon.

F.2 Changes to the encodings

The *OpenMath* version 2 standard still mandantes two encodings: xml and binary. The xml encoding in particular has been updated to reflect the latest development of xml and is now a full xml application. Version 2 encodings are backward compatible with version 1.1 encodings.

- both encodings have been updated to support the changes to the model of abstract objects described above.

- encodings support internal and external sharing of objects

- the xml encoding in version 2 is defined by a Relax NG schema and the mandated character -based grammar of version 1 has been removed, while the DTD has been relegated to an Appendix.

- an optional attribute defining the version of the encoding can be specified for the encoded object

F.3 Changes to Content Dictionaries

- In *OpenMath* version 2 Content Dictionaries are defined in terms of the abstract information content that needs to be specified for defining *OpenMath* symbols. The current implementation is thus just one possible encoding of this abstract model.

- The *CDUses* element is not part of this information model and has been made optional and deprecated in the reference encoding since it is trivial to extract its content automatically from the CD.

- A CD may now, optionally, define its uristub.

- A CD symbol definition may now, optionally, define its role.

Appendix G
Bibliography

[1] John A. Abbott, André van Leeuwen and A. Strotmann *OpenMath: Communicating Mathematical Information between Co-operating Agents in a Knowledge Network*, 1998.

[2] N. Borenstein and N Freed *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanism for Specifying and Describing the Format of Internet Message Bodies*, September 1993.

[3] O. Caprotti and A. M. Cohen *A Type System for OpenMath*, September 1998.

[4] Olga Caprotti and Arjeh M. Cohen *A Type System for OpenMath*, February 1999.

[5] S. Dalmas, M. Gaëtano and S. Watt *An OpenMath 1.0 Implementation*, 1997.

[6] J. Davenport *A Small OpenMath Type System*, April 1999.

[7] *IEEE Standard for binary Floating-Point Arithmetic*.

[8] IEEE *Std 1003.1 -2001 (Open Group Technical Standard, Issue 6), Standard for Information Technology-Portable Operating System Interface (POSIX)*, 2001.

[9] IETF *RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax*, August 1998.

[10] *ISO 7-bit coded character set for information interchange*, 1983.

[11] OASIS Committee Specification *RELAX NG Compact Syntax*, November 2002.

[12] OASIS Committee Specification *RELAX NG Specification*, December 2001.

[13] OpenMath Consortium *OpenMath Version 1 - Draft*, June 1998.

[14] Technical committee / subcommittee: JTC 1 *ISO 9660:1988 Information processing – Volume and File Structure of CDROM for Information Interchange*, 1988.

[15] Unicode Consortium *The Unicode Standard: Version 2.04.0*, 19962003.

[16] World Wide Web Consortium *XSD Schema Parts 1 & 2*, May 2001.

[17] World Wide Web Consortium *Namespaces in XML*, January 1999.

[18] World Wide Web Consortium *Extensible Markup Language XML 1.0*, February 1998.

[19] World Wide Web Consortium *Mathematical Markup Language (MathML) 2.0 Specification*, February 2001.

[20] World Wide Web Consortium *Extensible Stylesheet Language (XSL) Specification*, 21 Apr 1999.

[21] F. Yergeau *UTF-8, a transformation format of ISO 10646*, January 1998.