http://openmath.github.io

# OpenMath, A Technical Overview

## Introduction

OpenMath is a standard for representing mathematical data in as unambiguous a way as possible. It can be used to exchange mathematical objects between software packages or via email, or as a persistent data format in a database. It is tightly focussed on representing semantic information and is not intended to be used directly for presentation, although tools exist to facilitate this.

The original motivation for OpenMath came from the Computer Algebra community. Computer Algebra packages were getting bigger and more unwieldy, and it seemed reasonable to adopt a generic "plug and play" architecture to allow specialised programs to be used from general purpose environments. There were plenty of mechanisms for connecting software components together, but no common format for representing the underlying data objects. It quickly became clear that any standard had to be vendor-neutral and that objects encoded in OpenMath should not be too verbose. This has led to the design outlined below.

In 1998, the Worldwide Web Consortium (W3C) produced its first recommendation for the Extensible Markup Language (XML), intended to be a universal format for representing structured information on the worldwide web. It was swiftly followed by the first MathML recommendation which is an XML application oriented mainly towards the presentation (i.e. the rendering) of mathematical expressions.

The formal definition of OpenMath is contained within *The OpenMath Standard* and its accompanying documents, and the reader is referred there for more details.

## The OpenMath Architecture

The OpenMath representation of a mathematical structure is referred to as an *OpenMath object*. This is an abstract structure which is represented concretely via an *OpenMath encoding*. These encoded objects are what an OpenMath application would read and write, and in practice the OpenMath objects themselves almost never exist, except on paper. The advantage of this is that OpenMath is not tied to any one underlying mechanism: in the past we have used functional, SGML and binary encodings. The current favourite is XML, as described below, and we will tend to use XML notation when describing OpenMath objects (even though strictly speaking the XML representation is an encoding).

## OpenMath Objects

Formally, an OpenMath object is a labelled tree whose leaves are the *basic* OpenMath objects integers, IEEE double precision floats, unicode strings, byte arrays, variables or symbols. Of these, symbols are the most interesting since they consist of a name and a reference to a definition in an external document called a *content dictionary* (or CD). Using XML notation where the element name `OMS` indicates an OpenMath symbol, the following: `<OMS name="sin" cd="transc1"/>` represents the usual sine function, as defined in the CD "transc1". A basic OpenMath object is an OpenMath object, although its XML representation will be:

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

OpenMath objects can be built up recursively in a number of ways. The simplest is function application, for example the expression *sin*(*x*) can be represented by the XML:

```
<OMOBJ>
  <OMA>
    <OMS name="sin" cd="transc1"/>
    <OMV name="x"/>
  </OMA>
```

```
</OMOBJ>
```

where `OMV` introduces a variable and `OMA` is the application element. Another straightforward method is *attribution* which as the name suggests can be used to add additional information (for example “the AXIOM command which generated me was ...”) to an object without altering its fundamental meaning. More interesting are *binding* objects which are used to represent an expression containing bound variables, for example:

```
<OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="int"/>
    <OMS cd="transc1" name="sin"/>
  </OMA>
</OMOBJ>
```

represents the integral of the *sin* function, but the encoding:

```
<OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="int"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR> <OMV name="x"/> </OMBVAR>
      <OMA>
        <OMS name="sin" cd="transc1"/>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

represents ∫sin(*x*)dx. This may appear overly complicated but it is useful, for example when searching in a database for expressions which match ∫sin(*y*)dy . The definition of a symbol in the CD specifies whether or not it may be used to bind variables, which is why cannot be used as a binding symbol.

The final kind of OpenMath object is an *error* which is built up from a symbol describing the error and a sequence of OpenMath objects. For example:

```
<OMOBJ>
  <OME>
    <OMS name="unexpected_symbol" cd="error1">
    <OMS name="sine" cd="transc1">
  </OME>
</OMOBJ>
```

represents the error which might be generated when an application sees a symbol it doesn’t recognise from a CD it thought it knew about.

## OpenMath Encodings

We have already seen some examples of the XML encoding, but it is by no means the only encoding. In the past there was a functional encoding (which looked like Lisp) and an SGML encoding which evolved into the current XML. Both of these are now obsolete, but there is still a binary encoding described in the standard , which is much more compact than the XML one.

In fact the XML encoding is not 100% XML. When XML was in its infancy the developers of OpenMath realised that it might become significant and decided to add some XML-like features to the SGML encoding so that an an OpenMath object could be encoded as valid XML. Thus it is currently the case that any well-formed OpenMath object encoded using the XML encoding as described in the standard is a valid XML document. However, if one uses standard XML tools to generate an OpenMath object in the XML encoding from the DTD given in chapter 4 of the standard, it is possible that the result will not be valid OpenMath, although in practice this is highly unlikely. To cover all the possibilities allowed by XML would make it much more complicated to write an application to read any OpenMath object from scratch. Whether to adopt XML completely remains a hot topic of debate within the OpenMath community!

Generally speaking, it is not intended that the existing encodings should be readable by a human user or writable by hand. It is desirable that they be compact and it is also desirable that they be linear, but neither of these is a requirement. It is a property of encodings that it is possible to convert between them with no loss of information.

## Content Dictionaries

Content Dictionaries (or CDs for short) are the most important, and the most interesting, aspect of OpenMath because they define the meaning of the objects being transmitted. A CD is a collection of related symbols and their definitions, encoded in an XML format. Defining the meaning of a symbol is not a trivial task, and even referring to well-known references can be fraught with pitfalls Formal definitions and properties can be very useful but time-consuming to produce and verbose, not to mention difficult to get right. A symbol definition in an OpenMath CD consists of the following pieces of information:

- the symbol name;
- a description in plain text;
- optionally, a set of this symbol's properties in plain text (*Commented Mathematical Properties*, or *CMPs*);
- optionally, a set of this symbol's properties encoded in OpenMath (*Formal Mathematical Properties*, or *FMPs*);
- optionally, one or more examples of its use (encoded in OpenMath).

In practice the CMPs and FMPs can come as pairs, and often serve in the place of examples.

A very simple instance of a CD definition is:

```
<CDDefinition>
<Name> log </Name>

<Description>
This symbol represents a binary log function; the first argument is
the base, to which the second argument is log'ed.
It is defined in Abramowitz and Stegun, Handbook of Mathematical
Functions, section 4.1
</Description>
<CMP>
  a^b = c implies log_a c = b
</CMP>
<FMP>
  <OMOBJ>
    <OMA>
      <OMS cd="logic1" name="implies"/>
      <OMA>
        <OMS cd="relation1" name="eq"/>

        <OMA>
          <OMS cd="arith1" name="power"/>
          <OMV name="a"/>
          <OMV name="b"/>
        </OMA>
        <OMV name="c"/>
      </OMA>
      <OMA>
        <OMS cd="relation1" name="eq"/>

        <OMA>
          <OMS cd="transc1" name="log"/>
          <OMV name="a"/>
          <OMV name="c"/>
        </OMA>
        <OMV name="b"/>
      </OMA>
    </OMA>
  </OMOBJ>
</FMP>

<Example>
log 100 to base 10 (which is 2).
<OMOBJ>
  <OMA>
    <OMS cd="transc1" name="log"/>
    <OMF dec="10"/>
    <OMF dec="100"/>
  </OMA>
</OMOBJ>
</Example>

</CDDefinition>
```

This provides a symbol to represent the *log* function by giving a pointer to a standard reference book. It provides the property that:

$$a^b=c \rightarrow \log_a(c)=b$$

both as plain text and as OpenMath, and also gives an example of how the symbol is used.

CDs usually consist of related symbols and collections of related CDs can be grouped together, for convenience, as *CD Groups*. One very important CD Group is that corresponding to the content part of MathML. The set of CDs produced by the OpenMath Society can be browsed here.

It is possible to associate extra information with CDs, in particular type information. Since there are many type systems available, each of which has its own strengths and advocates, the OpenMath community does not mandate any single system. Simple signatures can be encoded using the Simple Type System, while more formal definitions are possible using the Extended Calculus of Constructorss. Other associated information can include style sheets for rendering OpenMath symbols in MathML, and mathematical definitions to be used by formal logic systems.

Given the evolutionary nature of mathematics, it is clear that the set of CDs should be forever growing and never complete. Currently there are CDs for high-school mathematics, linear algebra, polynomials and group theory to name a few, and new contributions are always welcome. There is no requirement that applications use the standard set of CDs and it is often very useful to design a “private” CD for a specific purpose.

## OpenMath in Action

There is no definitive way in which OpenMath should be used, as the protocol has been designed to be as flexible as possible. Nevertheless many OpenMath applications share common characteristics which we shall discuss here.

Suppose that we wish to have two applications communicating by sending OpenMath objects to each other, e.g. a client program and a computational server. It is unlikely that the internal data structures used by the applications will be OpenMath, and so translation between the internal representations and OpenMath (almost certainly OpenMath encodings rather than objects) will have to take place. The piece of software which does this is usually referred to as a *phrase-book*.

It is possible to write a generic phrase-book which can handle any piece of OpenMath, but applications where this makes sense are few and far between. In practice an OpenMath phrase book will usually only handle a fixed set of CDs (and hence a fixed set of symbols). What “handle” means will vary from case to case: a computer algebra system will usually try and evaluate its input and return a result or an error, while a typesetter will print its input according to some rendering rules and not return anything. OpenMath carefully avoids defining what the “right” behaviour is in a given circumstance, and leaves that up to the phrase-book writer. Indeed it is quite possible that a piece of software could have multiple phrase-books associated with it for different purposes. OpenMath symbols should not be regarded as verbs since they are used to construct objects rather than to send commands, and the presence of both nouns and verbs in a CD (e.g. “integral” and “integrate”) is strongly discouraged.

Writing a phrase-book may be non-trivial, and requires an understanding of the semantics of the underlying software. An OpenMath object may not map directly into a private object and vice-versa, for example in some systems a rational number might have to be represented by a float, or a sparse matrix by a dense one.

The OpenMath standard includes a section on compliance, which describes the behaviour of an OpenMath application when certain errors occur. It also insists that all compliant software has the capability to use the XML encoding, to guarantee a degree of interoperability. This is an area where the standard is expected to evolve as more OpenMath applications become available.

A list of applications of OpenMath can be found in the software & tools section of this website.