# On Writing OpenMath Content Dictionaries

James H. Davenport*
Dept. Mathematical Sciences
University of Bath
Bath BA2 7AY
England
jhd@maths.bath.ac.uk

**Abstract**

This paper is based on various discussion with the OpenMath Consortium, and recently at the University of Western Ontario. All errors are the author's. Many helpful suggestions have been made, particularly by Dr. Dewar and Dr. Naylor.

This paper outlines some of the issues that affect authors of OpenMath Content Dictionaries, and their associated Small Type System [4] files.

## 1 Introduction

This paper addresses the following questions about the writing of OpenMath Content Dictionaries (CDs), and associated Small Type System (STS) [4] files.

1. Why write a Content Dictionary?
2. What should I look at before/while writing a Content Dictionary?
3. What should I bear in mind while writing a Content Dictionary?
4. What is approval for a Content Dictionary, and how do I get a Content Dictionary approved?

### 1.1 Content Dictionary Groups

A Content Dictionary Group is a file (normally with extension `.cdg`) which specifies a grouping of CDs for a logical purpose. For example, the MathML group is a group of CDs whose symbols are equivalent to the symbols of Content MathML [12, Appendix C].

A CD can be in more than one CD group. For example, `setname1.ocd` is in the MathML CD group, since it contains symbols (**Z** etc.) which are in content MathML. It is also in the `setname.cdg` group, which also contains `setname2.ocd`, which has various set names, such as $\mathcal{P}$, which are not in content MathML.

## 2 Why write a CD?

A Content Dictionary is a fundamental concept of OpenMath. The symbols contained in a CD form the mechanism by which OpenMath achieves its goal of being an "an extensible framework for exchanging semantically-rich representations of mathematical objects". So the motivation for writing a CD is that there is some semantics that one wishes to exchange. The consequences of this motivation are the following.

1. There must be a "new" piece of semantic information to convey.
2. It must be possible to write down informally (the "commented mathematical properties" or CMPs) the semantics that the author of the CD intends to convey. It should also be possible to write Formal Mathematical Properties (FMPs), but this is not necessary, and is clearly impossible for all of mathematics. It may be that some of the new items can be defined in terms of others, even if not everything can be defined formally.
3. There must be a motivation for wishing to convey it. One such motivation would be that two algebra systems wished to communicate objects with these semantics, but this is far from the only motivation. Databases might contain these objects (consider the data described in [2]), or we may wish to search in papers containing such items in formulae.

## 3 An example: multisets

MathML [12] defines the concept of `set`, and says that it can have a `type` attribute of `normal`[1] or `multiset`. OpenMath has a `set1.ocd` and corresponding `set1.sts` (available as [9]) which define the semantics for ordinary sets. Though they do not explicitly say so, it is the case that $A \cup A = A$. Since OpenMath does not have[2] the same sort of attribute concept that MathML has, we need a way to encode multisets in OpenMath.

The following possibilities exist.

1. Add a `multiset` constructor to `set1.ocd`
2. Define a new CD (probably called `multiset1`) with the same operators as `set1` (except that `set` would probably be called `multiset`) but different semantics.
3. Define a new CD (probably called `multiset1`) with operators such as `multiset-union` etc.

*Supported by the OpenMath Esprit Project 24.696.

[1] Presumably the default, though this is not made explicit.

[2] For good reasons: it's only possible in XML to have fixed attributes of the MathML kind, which would conflict with the extensibility of OpenMath.

The first has some drawbacks.

- The semantics are not the same, so one needs to say something like "If $A$ is a set (rather than a multiset) then $A \cup A = A$". Lest this seem trivial, consider the fact that
$$A \cup B = (A \setminus B) \cup (A \cap B) \cup (B \setminus A)$$
is true for sets but false for multisets: if $A = \{1,1,2\}$ and $B = \{1,2,2\}$, the the left-hand side is $\{1,1,1,2,2,2\}$, but the right-hand side is $\{1,1,2,2\}$.
- It is then impossible to distinguish between the operation of union on sets and multisets. Some texts use different symbols for the two, thus allowing a text to write "$A \cup A = A$, but $A \sqcup A \neq A$". It is also possible that an algorithm might wish to consider both.

Hence the choice lies between the second and third. This is a matter of preference, but it seems that the general OpenMath convention[3] has been to choose the second rather than the third. In the current context, one can see that

```
<OMS name="multiset-union" cd="multiset1"/>
```

seems somewhat redundant compared with

```
<OMS name="union" cd="multiset1"/>
```

Examples of a CD and the associated STS file constructed on this principle are given in the full version [5], also [9].

## 4 What to Look at

The key document is the formal OpenMath standard [8]. Clearly this document is important, as is the description of the Small Type System [4]. Existing CDs, particularly the draft official ones at [9], are a useful source of layout[4], but also of examples of how things can be described. In particular, when looking at writing `multiset1.ocd`, the author looked carefully at `set1.ocd`. However, the reader will notice that not all the examples were blindly copied: some have been changed to examples more appropriate for the case at hand.

In the field in which you are working, there may be some standard reference works, e.g. [1] in the area of special functions. These should certainly be consulted, but it should be borne in mind that such works, however famous, may not be complete (see [3] for examples). Equally, there may be standard software systems, and it would make sense to look at them first. However, one should not expect uniformity here. The following table shows what happens on an apparently simple example: the definition of $\operatorname{arccot}(-1)$.

| | | | |
|---|---|---|---|
| [1] | 1st printing | $3\pi/4$ | inconsistent |
| [1] | 9th printing | $-\pi/4$ | |
| [6] | 5th edition | ? | inconsistent |
| [13] | 30th edition | $3\pi/4$ | inconsistent |
| Maple | V release 5 | $3\pi/4$ | |
| Axiom | 2.1 | $3\pi/4$ | |
| Mathematica | [11] | $-\pi/4$ | |
| Reduce | 3.4.1 | $-\pi/4$ | in floating point |
| Matlab | 5.3.0 | $-\pi/4$ | in floating point |
| Matlab | 5.3.0 | $3\pi/4$ | symbolic toolbox |

[3]For example, the operator in `arith2` is called `times` not `commutative-times`. However, this is not an invariable rule, since the proposed CD for multi-valued inverse functions uses Log for the multi-valued equivalent of log, since this is the common mathematical convention.

[4]The formal rules for an OpenMath CD are given in [7].

## 5 What to Bear in Mind

The key things to bear in mind while writing a CD are the following.

1. OpenMath is about semantics, rather than the elegance of rendering. There are many alternative ways to determine how something is rendered, but this is the job of MathML [12], particularly its presentation mode, rather than of OpenMath.

   One example will illustrate this point. A colleague commented as follows. "It occurred to me whilst writing that some OpenMath phrases like '$a, b, c \in \mathbf{Z}$' seem to occur quite frequently in mathematics. At the moment the only way to encode this in OpenMath is the following:

```
<OMA>
  <OMS cd="logic1" name="and"/>
  <OMA>
    <OMS cd="set1" name="in"/>
    <OMV name="a"/>
    <OMS cd="setname1" name="Z"/>
  </OMA>
  <OMA>
    <OMS cd="set1" name="in"/>
    <OMV name="b"/>
    <OMS cd="setname1" name="Z"/>
  </OMA>
  <OMA>
    <OMS cd="set1" name="in"/>
    <OMV name="c"/>
    <OMS cd="setname1" name="Z"/>
  </OMA>
</OMA>
```

   Surely it would be much neater if instead 'in' was made $(n+1)$-ary, and we could say something like:

```
<OMA>
  <OMS cd="set1" name="in"/>
  <OMV name="a"/>
  <OMV name="b"/>
  <OMV name="c"/>
  <OMS cd="setname1" name="Z"/>
</OMA>
```

   ...".

   We note, however, that no "new" semantics were involved: indeed the fact that an existing representation in OpenMath was quoted essentially proves this. This led to the analysis in Table 1, and a general principle: MathML (Content) and OpenMath will aim for simplicity in the core language at the cost of requiring a more complex translation into "good" written mathematics.

   At this point, the reader may well object that this principle has not always been followed in the most basic OpenMath Content Dictionaries. For example, we have `not` and `in`, so why `notin`, since the justification for replacing

```
<OMA>
  <OMS cd="logic1" name="not"/>
```

| Area | binary $\in$ (status quo) | $n$-ary $\in$ |
|---|---|---|
| Translation from TEX etc. | An easy transformation | nothing needed |
| Formal Reasoning | Nothing special | An extra rule needs to be added: in practice most systems will probably remove $n$-ary $\in$ |
| Generating 'quality' human output | Need a rule to flatten binary $\in$ to $n$-ary | Might not need such a rule, but that's not certain |
| Humans reading or writing MathML or OpenMath directly | Frankly, they have enough problems that who cares? | |

Table 1: Binary or $n$-ary $\in$?

```
    <OMA>
      <OMS cd="set1" name="in"/>
      <OMV name="a"/>
      <OMS cd="setname1" name="Z"/>
    </OMA>
  </OMA>
```

by

```
  <OMA>
    <OMS cd="set1" name="notin"/>
    <OMV name="a"/>
    <OMS cd="setname1" name="Z"/>
  </OMA>
```

seems to be the same as that rejected above.

The answer here is that the "core" CDs of OpenMath are required to possess a natural mapping onto MathML, and this was, unfortunately, a higher priority than this general principle *as far as these CDs were concerned*. This may sound like a case of "do as I say, not as I do", but the OpenMath design community have tried to apply this principle whenever MathML compatibility did not override the rule.

2. It is important to define the mathematically natural concept, rather than the computationally natural one. This may sound obvious, but we will give an illustration: the Bessel functions. Consider the Bessel function ($J_\nu(z)$ for simplicity: the same points are true for other functions). This could be defined, as it is in all numerical libraries, as a function $(\mathbf{C} \times \mathbf{C}) \to \mathbf{C}$, i.e. (in STS)

```
<OMA>
  <OMS name="mapsto" cd="sts"/>OMV
  <OMS cd="sts" name="NumericalValue"/>
  <OMS cd="sts" name="NumericalValue"/>
  <OMS cd="sts" name="NumericalValue"/>
</OMA>
```

but it makes more sense to define it as $\mathbf{C} \to (\mathbf{C} \to \mathbf{C})$, i.e. (in STS)

```
<OMA>
  <OMS name="mapsto" cd="sts"/>
  <OMS cd="sts" name="NumericalValue"/>
  <OMA>
    <OMS name="mapsto" cd="sts"/>
    <OMS cd="sts" name="NumericalValue"/>
    <OMS cd="sts" name="NumericalValue"/>
  </OMA>
</OMA>
```

so that we can say that $J(\nu)$ satisfies Bessel's equation, rather than having to say that $\lambda x.J(\nu, x)$ satisfies Bessel's equation.

3. Do not pun, or, harder, perpetrate general puns that you may not even have noticed. If there are different meanings for a piece of notation, then there should be different symbols. Two classic examples from the MathML CD group are the existence of both `minus` and `unary_minus` in `arith1.ocd`, and the existence of both `int` and `defint` in `calculus1.ocd`.

4. Use CDgroups, and do not believe that a single CD needs to solve everything. CDs can be split thematically: for example, rather than having one CD of special functions, one could have one for Bessel (and, possibly, Bessel-like) functions, one for special integrals, such as the sine-integral and, possibly erf, and so on. All these could be in one CDgroup of special functions. Doing this split has the advantage that

```
<OMS name="J" cd="Bessel"/>
```

is slightly easier to write (not that direct writing of OpenMath is to be encouraged) than

```
<OMS name="BesselJ" cd="Special1"/>
```

and, more seriously, is more likely to render as $J$ without special processing.

CDs can also be split by depth, so `group1.ocd` might contain some relatively common group-theoretic concepts, whereas `group2.ocd` etc. might contain more abstruse ones.

5. The choice of symbol name is important. As seen in the previous paragraph, it is possible to have brief symbol names if they are unambiguous in the context of that CD.

One perennial question is the use of upper/lower case in a symbol name. In general, lower case is used, except that multi-word names in which no other punctuation is used generally start the second (and subsequent) words with a capital to make reading the symbol name easier. Initial capitals are normally used in the following settings.

   (a) Well-known set names: Z, Q etc. This rule is deemed to include similar names such as NumericalValue and Object from `sts.ocd`.

(b) Well-known functions which normally take a capital: J etc., and NaN, also, as is usual in mathematics, the multi-valued elementaries: Log and Arcsin etc. This rule also covers `LaTeX_encoding` etc.

(c) Type constructors like `SigmaType` and `PiType`, named for capital Greek letters, and other constructors in the ECC world, such as `Setoid`.

(d) meta.ocd is exempt, since the cases should match those in OpenMath keywords.

(e) General abbreviations, such as CD itself, also DMP (for distributed multivariate polynomial).

## 6 Good CD-writing style

There are various guidelines that make reading a CD easier, for humans and for machines. We list some of them here: the list will doubtless evolve as the community gains more experience in writing, and criticising, CDs.

1. Give examples. Every symbol defined in a CD should have at least one example, either as such (i.e. `<Example>`), or via an FMP which uses it. Where there are significantly different uses of a symbol (see `set` in `set1.ocd` for an example), multiple examples are probably appropriate. In general, FMPs are more useful than examples, since formal systems can also read the FMPs, whereas there are no additional semantics associated with examples. However, examples can illustrate uses (even idioms) which don't have a place in FMPs, so there can be no blind rule "convert all examples into FMPs".

2. If an FMP is given, then the equivalent (English) CMP should also be given. If possible, then the converse should also be done. CMPs should be written in English, and preferably avoiding abbreviations (or special notation, e.g. TeX)[5], e.g. "if a and b are real numbers ..." rather than "a,b in R ...". Note that symbols in CDs represent concepts, they do not represent, or perform, operations.

3. FMPs should be as comprehensive as reasonable (clearly not all theorems of set theory could be given in `set1.ocd`, though). Examples tend, on the other hand, to aim for brevity.

4. The `CDUses` field, which lists all the other CDs which have symbols occurring in this CD (i.e. in examples and FMPs) should be correct. There are tools to assist with this.

5. While it would be naïve to expect a CD to be completely self-contained, some restraint is necessary. In particular, a CD should not use CDs which are less "official" than it itself is. Ideally CDs in a given CD group would refer to each other and official CDs only.

6. The corresponding STS file [4] should be written. These signatures should be as helpful as possible, and not say, for example, Object→Object simply because the writer is lazy.

[5]It is hoped (at least by the author) that, as MathML becomes more widely adopted and conventions for mixing different XML languages become established, the DTD for CDs will allow MathML descriptions in CDs.

## 7 The Approval Process

The OpenMath Society (`http://www.openmath.org`) is the official guardian of approved OpenMath CDs. These can be referred to by their URL on the OpenMath Web site.

Currently (until 30.9.2000) the Esprit OpenMath project is suggesting CDs to the OpenMath Society. Professor Davenport (`jhd@maths.bath.ac.uk`) is the coordinator of this process, and suggestions should be made to the Esprit project (`openmath-project@nag.co.uk`) or directly to him.

After this period, the OpenMath Society has formulated a procedure for submitting CDs. The full details are at [10], but in brief the CD is submitted to any member of the Executive Committee: `http://www.nag.co.uk/projects/OpenMath/omsoc/society/board.html`

## References


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

[2] Conway,J.H., Hulpke,A. & McKay,J., On Transitive Permutation Groups. LMS J. Computation and Math. **1** (1998) pp. 1–8.

[3] Corless,R.M., Davenport,J.H., Jeffrey,D.J. & Watt,S.M., "According to Abramowitz and Stegun". This issue of the SIGSAM Bulletin.

[4] Davenport,J.H., A Small OpenMath Type System. OpenMath Esprit Project Deliverable 1.3.2c. `http://www.nag.co.uk/projects/OpenMath/omstd/sts.ps.gz`. See also this issue of the SIGSAM Bulletin.

[5] Davenport,J.H., On Writing OpenMath Content Dictionaries. OpenMath Esprit Project Deliverable 1.3.2c. `http://openmath.nag.co.uk/openmath/deliverabls/d142/d142.tex`

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

[7] The OpenMath Consortium. A DTD for OpenMath Content Dictionaries. `http://www.nag.co.uk/projects/OpenMath/omstd/omcd.dtd`.

[8] The OpenMath Consortium. Draft OpenMath Standard. `http://www.nag.co.uk/projects/OpenMath/omstd`.

[9] The OpenMath Consortium. Draft Content Dictionaries. `http://www.nag.co.uk/projects/OpenMath/corecd/`.

[10] The OpenMath Society. Content Dictionary Acceptance Procedures. `http://www.nag.co.uk/projects/OpenMath/omsoc/standard/cdacc.html`.

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

[12] World-Wide Web Consortium, Mathematical Markup Language (MathML [tm]) 2.0 Draft Specification. W3C Draft, revision of 11 February 2000. `http://www.w3.org/TR/2000/WD-MathML2-20000211`.

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