Unifying Math Ontologies: A tale of two standards

James H. Davenport[1] and Michael Kohlhase[2]

[1] Department of Computer Science
University of Bath, Bath BA2 7AY, United Kingdom
J.H.Davenport@bath.ac.uk

[2] School of Engineering & Science, Jacobs University Bremen
Campus Ring 12, D-28759 Bremen, Germany
m.kohlhase@jacobs-university.de

Abstract. One of the fundamental and seemingly simple aims of mathematical knowledge management (MKM) is to develop and standardize formats that allow to "represent the meaning of the objects of mathematics". The open formats OpenMath and MathML address this from a content markup perspective, but differ subtly in syntax, rigor, and structural viewpoints (notably over calculus). To avoid fragmentation and smooth out interoperability obstacles, effort is under way to align them into a joint format OpenMath/MathML3. We illustrate the conceptual and practical issues that come up in such an alignment by looking at three main areas: bound variables and conditions, calculus (which relates to the previous) and "lifted" n-ary operators.

Whenever anyone says "you know what I mean", you can be pretty sure that he *does not know what he means, for if he did, he would tell you.*
— Anon.

1 Introduction

One of the fundamental and seemingly simple aims of mathematical knowledge management (MKM) is to develop and standardize representation formats that allow one to specify the meaning of the objects and documents of mathematics. The open formats OpenMath and MathML address the key sub-problem of representing mathematical objects from a content markup perspective: mathematical objects are represented as expression trees. As the formats were developed by different communities with different goals, they differ subtly in syntax, rigor, and structural viewpoints (notably over calculus). This has caused double developments, interoperability problems, and confusion in developers, system vendors, and users of mathematical software systems and has considerably weakened the uptake of MKM methods. The efforts to mitigate the interoperability problem by establishing translations between the formats have done more to unearth subtle problems than to completely solve them in the past.

In this paper we report on an ongoing effort of the W3C MathML Working group and members of the OpenMath Society to merge the ontologies[3] on which the OpenMath and MathML formats are based and thus align the formats, so that they only differ in their concrete XML encodings. This task proves to be harder than might initially be expected. We explain why, motivated by a study of four areas (which in fact turn out to be inter-related):

1. constructions with bound variables;
2. the `<condition>` element of MathML;
3. the different handling of calculus-related operations in the two;
4. the "lifting" of n-ary operators, such as $+$ to $\sum$.

This paper is a short version of [DK09c], which contains the details of the constructions. OpenMath-specific details of the proposals are in [DK09b,DK09a].

2 OpenMath and MathML

We will now recap the two formats focusing on their provenance and representational assumptions and then sketch the measures taken for aligning the languages. Sections 3, 4, 6, and 7 will detail the problem areas identified above. The first two leading to an extension proposal for OpenMath Objects and strict content MathML in Section 5, which is evaluated in the latter two. Section 7 concludes the paper.

2.1 MathML

MathML is an XML-based language for capturing mathematical the presentation, structure and content of mathematical formulae, so that they can be served, received, and processed on the World Wide Web. Thus the goal of MathML is to provide a similar functionality that HTML has for text. The present recommended version of MathML format is MathML 2 (second edition) of October 2003 [Con03]. MathML 1 had been recommended in April 1998 and revised as MathML 1.01 in July 1999.

MathML, starting from version 1.0, had a split into **presentation MathML**, describing what mathematics "looked like"[4], and **content MathML**, describing what it "meant". In this paper we will concentrate on content MathML, since

[3] Here we use the word "ontology" in its general, philosophical meaning as the study of the existence of objects, their categories and relations amongst each other, and not in the Semantic Web usage, where it is restricted to formal systems with tractable inference properties (description logics). Note furthermore that we are speaking as much about a "meta-ontology" of mathematical representation concepts as about "domain ontologies" that describe the mathematical concepts themselves. Having made this distinction, we will conveniently gloss over it in the rest of the paper.

[4] Which could include "sounded like" (for aural rendering) or "felt like" (e.g. for Braille), and MathML included a range of symbols such as `⁢` to help with this task.

the role of presentation MathML as a high-level presentation format for Math on the Web is (largely) uncontested. MathML's Content markup has ambitious goals:

> *The intent of the content markup in the Mathematical Markup Language is to provide an explicit encoding of the* underlying mathematical structure *of an expression, rather than any particular rendering for the expression.* [Con03, section 4.1.1]

This mandate is met in MathML 1/2 by representing mathematical formulae as XML expression trees that follow the applicative structure of operators and their arguments: function application is represented by the `apply` elements where the first child is interpreted as the operator and the remaining children as their arguments. MathML2 supplies about 90 elements for mathematical operators. The language has a fairly limited vision of what might be in "content":

> *The base set of content elements are chosen to be adequate for simple coding of most of the formulas used from kindergarten to the end of high school in the United States, and probably beyond through the first two years of college, that is up to A-Level or Baccalaureate level in Europe. [Con03, 4.1.2]*

This is often referred to as the **K-14 fragment** of mathematics. Since Version 2, MathML does have an extension mechanism via the `csymbol` elements and their `definitionURL` attributes, but this was rarely used except to achieve some form of OpenMath interoperability, or for proprietary extensions (e.g. Maple).

MathML tries to cater to the prevalent representational practices of mathematicians, and provides a good dozen structural XML elements for special constructions, e.g. set, interval and matrix constructors, and allows to "lift" various associative operators to "big operators" acting on sets and sequences simply by associating them by bound variables and possibly qualifier elements to specify the domain of application.

The MathML approach to specifying the "meaning" of expression trees largely follows a "you know what I mean" approach that alludes to a perceived consensus among mathematical practitioners on the K-14 fragment. The meaning of a construction is alluded to via examples rather than defined rigorously, intending to be "formal enough" to cover "*a large number of applications*" [Con03, 4.1.2], while remaining flexible enough not to preclude too many.

2.2 OpenMath

OpenMath is a standard for the representation and communication of mathematical objects. It has similar goals to content MathML and focuses on encoding the meaning of objects rather than visual representations to allow the free exchange of mathematical objects between software systems and human beings. OpenMath has been developed in a long series of workshops and (mostly European) research projects that began in 1993 and continues through today. The

OpenMath 1.0 and 2.0 Standards were released by the OpenMath Society in February 2000 and June 2004. OpenMath 1 fixed the basic language architecture, while OpenMath2 brought better XML integration, structure sharing and separated the notion of OpenMath Content Dictionaries from their encoding.

Like content MathML, OpenMath represents mathematical formulae as expression trees, but concentrates on an extensible framework built on a minimal structural core language with a well-defined extension mechanism. Where MathML supplies more than a dozen elements for special constructions, OpenMath only supplies concepts for function application (`OMA`), binding constructions (`OMBIND`; MathML2 lacks an analogous element and simply uses `apply` with bound variables, hence the (inferred) Rule 1.). Where MathML provides close to 100 elements for the K-14 fragment, OpenMath gets by with only an `OMS` element that identifies symbols by pointing to declarations in an open-ended set of Content Dictionaries (see below).

An OpenMath Content Dictionary (CD) is a document that declares names (OpenMath "symbols") for basic mathematical concepts and objects. CDs act as the unique points of reference for OpenMath symbols (and their encodings the `OMS` elements) and thus supply a notion of context that situates and disambiguates OpenMath expression trees. To maximize modularity and reuse, a CD typically contains a relatively small collection of definitions for closely related concepts. The OpenMath Society maintains a large set of public CDs, including CDs for all pre-defined symbols in MathML 2. There is a process for contributing privately developed CDs to the OpenMath Society repository to facilitate discovery and reuse. OpenMath does not require CDs be publicly available, though in most situations the goals of semantic markup will be best served by referencing public CDs available to all user agents.

The fundamental difference to MathML is in terms of establishing meaning for mathematical objects. Rather than appealing to mathematical intuition, OpenMath defines a free algebra $\mathcal{O}$ of "OpenMath Objects" which acts as (initial) model for encodings of mathematical formulae. OpenMath Objects are essentially labeled trees, with α-conversion for binding structures and Currying for nested semantic annotations. Note that since $\mathcal{O}$ is initial it is essentially unique and identifies (in the sense of "declares to be the same") fewer objects than any other model. As a consequence two mathematical objects must be identical, if their OpenMath representations are, but may coincide, even if their representations are different. The OpenMath standard therefore considers OpenMath objects as primary citizens and views the "OpenMath XML encoding" as just an incidental design choice for an XML-based markup language. In fact OpenMath specifies another encoding: the "binary encoding" designed to be more space efficient at the cost of being less human-readable.

2.3 The OpenMath/MathML3 Alignment Process

Most of these differences between MathML and OpenMath can be traced to the different communities who developed these representation formats. MathML came out of the "HTML Math Module", an attempt to develop LaTeX-quality presentation of mathematical on the Web, something sorely missing from the otherwise very successful HTML. The guiding goal for OpenMath on the other hand was to develop an open interchange format among computer algebra systems, which resulted in a much stronger emphasis on the meaning of objects to make the exchange of sub-problems safe.

Even though interoperability between OpenMath and and MathML was always a strong desideratum for both communities, the two representation formats evolved independently and in line with the fundamental assumptions outlined in the two previous sections. Interoperability was attempted from the MathML side by integrating the `csymbol` element in MathML2 and specifying parallel markup, i.e. allowing OpenMath representations to be embedded into MathML with fine-grained cross-referencing. The OpenMath Society developed CDs with analogues for "all predefined operators" and specified the correspondence between expression trees in [CDD$^+$01]. Although 30 pages long, the fact that this document is still incomplete may serve as an indication that the problem is not trivial. As we will see below, mapping the MathML operators is not enough in the presence of different structural elements in the formats.

In June 2006 the W3C rechartered the MathML Working Group to produce a MathML 3 Recommendation, and the group identified the lack of regularity and specified meaning as a problem to be remedied in the charter period. The group decided to establish meaning for content MathML expressions based on OpenMath objects without losing backwards compatibility to content MathML2. In the end, content MathML was extended to incorporate concepts like binding structures and full semantic annotations from OpenMath and a structurally regular subset of the extended content MathML was identified that is isomorphic to OpenMath objects. This subset is called **strict content MathML** to contrast it to full content MathML that was seen to strike a more pragmatic balance between regularity and human readability. Full content MathML borrows the semantics from strict MathML by a mapping specified in the MathML3 specification that defines the meaning of non-strict (**pragmatic**) MathML expressions in terms of strict MathML equivalents. The division into two sub-languages serves a very important goal in standardization: to clarify and codify best (engineering) practices without breaking legitimate uses in legacy documents. In the current third version of MathML, the latter is a primary concern.

In June 2007, the OpenMath society chartered a group of members which includes the authors of this paper to work on version 3 of the OpenMath standard which would recognize content MathML3 as a legitimate OpenMath encoding, to help define the pragmatic to strict mapping MathML, and to provide the necessary CDs, which would be endorsed by the W3C Math Group and the OpenMath Society. The discussions and the resulting CDs are online in the SWiM Wiki [LGP08] [Lan09]

Subsequent sections describe the problem areas that came up during the work and needed to be circumnavigated.

MKM2009.tex 516 2009-03-12 16:08:30Z jhd

3 Set Constructors in MathML

With the K-14 scope discussed above, MathML found that it needed more sophisticated concepts, such as bound variables, to express the concepts that are manipulated *informally* at that level. One conspicuous example from K-14 is sets constructed by rules.

> *A typical use of a qualifier is to identify a bound variable through use of the bvar element [. . .] The* `condition` *element is used to place conditions on bound variables in other expressions. This allows MathML to define sets by rule, rather than enumeration, for example. The following markup, for instance, encodes the set* $\{x|x < 1\}$*:*

```
<set>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><lt/><ci>x</ci><cn>1</cn></apply>
  </condition>
</set>
```

[Con03, 4.2.1.8]

Here (with the benefit of a great deal of hindsight, it should be pointed out) we can see the start of the problem. What would we have meant if we had changed the second[5] `x` to `y`? We would, of course, have written the MathML equivalent of $\{x|y < 1\}$, and the MathML would be as meaningless as that set of symbols. We therefore deduce the following (undocumented) rule, which corresponds to OpenMath's formal rules for `OMBIND`.

Rule 1 (MathML) *Variables in* `bvar` *constructions 'bind' the corresponding variable occurrences in the scope of the parent of the* `bvar`*. However, the variable may (e.g.* $\forall$*) or may not (e.g.* $\frac{\mathrm{d}}{\mathrm{d}x}$*) be bound in the* λ*-calculus sense.*

Here the first problem of interpreting pragmatic MathML elements raises its ugly head. In OpenMath, we can represent the set[6] $\{x \in \mathbb{R}|x < 1\}$ by the representation

```
<OMOBJ version="2.0">
  <OMA>
    <OMS cd="set1" name="suchthat"/>
    <OMS cd="setname1" name="R"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR><OMV name="x"/></OMBVAR>
      <OMA>
        <OMS cd="relation1" name="lt"/>
```

[5] Changing both of them would have essentially been an α-conversion [Bar84, Definition 2.1.11], though MathML does not analyse the concept.

[6] Note that the OpenMath CDs require a larger set to be specified (to avoid Russell's paradox). It would not be a problem to provide a CD for what is often called "naïve set theory" that leaves out this safety device. However, such a system would have the same difficulties that the MathML above has: do we mean $(-\infty, 1)$ or $[0, 1)$, and is this a subset of $\mathbb{Z}$ or $\mathbb{R}$?

```
      <OMV name="x"/>
      <OMI> 1 </OMI>
    </OMA>
  </OMBIND>
 </OMA>
</OMOBJ>
```

This makes use of a binding construction (`OMBIND`) with a λ operator that constructs functions[7] from an expression with a bound variable. This kind of construction is standard in logical systems and λ-calculus, for which is is motivated as follows in a standard introductory textbook:

> *To motivate the λ-notation, consider the everyday mathematical expression '$x - y$'. This can be thought of as defining either a function f of x or g of y ... And there is need for a notation that gives f and g different names in some systematic way. In practice mathematicians usually avoid this need by various 'ad hoc' special notations, but these can get very clumsy when higher-order functions are involved.* [HS08, p. 1]

To achieve interoperability with OpenMath objects, MathML3 introduces the `bind` element in analogy to the OpenMath `OMBIND`. It could be argued that the "K–14" brief of MathML rules out higher-order functions, but in the example above we can see here the need, in a purely first-order case, to resort to "well, you know what I mean" without it. Extending MathML3 with a `bind` element that encodes an *OpenMath binding object* takes the guessing of Rule 1 out of MathML and makes the meaning unambiguous. The MathML3 specification does however need to specify the strict content MathML equivalent for the MathML2 example above in order to give it an OpenMath Object semantics.

4 Calculus Issues

MathML and OpenMath have rather different views of calculus, which goes back to a fundamental duality in mathematics. These can, simplistically, be regarded as:

- what one learned in calculus, which *we* will write as $D_{\epsilon\delta}$: the "differentiation of ϵ–δ analysis". Also $\frac{\mathrm{d}}{\mathrm{d}_{\epsilon\delta} x}$, and its inverse ${}_{\epsilon\delta}\int$;
- what is taught in differential algebra, which *we* will write as D_{DA}: the "differentiation of differential algebra". Also $\frac{\mathrm{d}}{\mathrm{d}_{\mathrm{DA}} x}$, and its inverse ${}_{\mathrm{DA}}\int$.

Roughly speaking, the MathML encoding corresponds more closely to $D_{\epsilon\delta}$ and the OpenMath one to D_{DA}. If we were to look at the derivative of x^2 as in Figure 1, we might be tempted to see only trivial syntactic differences: In the MathML encoding we see a differential operator that *constructs a function from an expression with a bound variable*[8] declared by a `bvar` element. The

[7] Here we also make use of the duality between sets and Boolean-valued functions that are their characteristic functions

[8] With the insights from the last section, MathML3 would probably use a `bind` element, emphasizing the role of the differentiation operator as a function constructor.

<**apply**>	<**OMA**>
<**diff**/>	<**OMS** cd="calculus1" "name="diff"/>
	<OMBIND>
	<OMS cd="fns1" name="lambda"/>
<**bvar**><**ci**>x</**ci**></**bvar**>	<OMBVAR><OMV name="x"/></OMBVAR>
<**apply**>	<OMA>
<**power**/>	<OMS cd="arith1" name="power"/>
<**ci**>x</**ci**>	<OMV name="x"/>
<**cn**>2</**cn**>	<OMI>2</OMI>
</**apply**>	</OMA>
	</OMBIND>
</**apply**>	</OMA>

Fig. 1. MathML2 and OpenMath2 differentiation compared

OpenMath encoding sees the differential operator as a functional that transforms one function (the square function) into another (its derivative). It is possible to do this without any variables, as in $\sin' = \cos$. Given the history of the two standards, this difference of encoding is not surprising, since D_{DA} is what computer algebra systems do (and what humans do, most of the time, even while interpreting the symbols as $D_{\epsilon\delta}$), whereas human beings generally think they are doing $D_{\epsilon\delta}$ and communicate mathematics that way.

For partial differentiation we see the same general picture, but the concrete representations drift further apart: For $\frac{d^{m+n}}{dx^m\,dy^n} f(x,y)$, MathML would use

```
<apply>
  <partialdiff/>
  <bvar><ci>x</ci><degree><ci>m</ci></degree></bvar>
  <bvar><ci>y</ci><degree><ci>n</ci></degree></bvar>
  <degree><apply><plus/><ci>m</ci><ci>n</ci></apply></degree>
  <apply><ci type="function">f</ci><ci>x</ci><ci>y</ci></apply>
</apply>
```

using `degree` qualifiers inside the `bvar` elements for the orders of partial differentiations and a `degree` qualifier outside for the total degree. The following representation is proposed in [CDD+01]:

```
<OMA>
  <OMS cd="calculus1" name="partialdiff"/>
  <OMA>
    <OMS cd="list1" name="list">
    <OMV name="m"/>
    <OMV name="m"/>
  </OMA>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR><OMV name="x"/><OMV name="y"/></OMBVAR>
    <OMA><OMV name="f"><OMV name="x"/><OMV name="y"/></OMA>
  </OMBIND>
</OMA>
```

For the problems caused by wishing to represent $\frac{d^k}{dx^m\,dy^n} f(x,y)$, see [Koh08] and the proposed solution in [DK09a].

Integration is even more problematic than differentiation. MathML interprets integration as an operator on expressions in one bound variable and presents as paradigmatic examples the three expressions below, which differ in which ways the bound variables are handled.

a: $\int_0^a f(x)dx$	b: $\int_{x\in D} f(x)dx$	c: $\int_D f(x)dx$
<**apply**> <**int**/> <**bvar**> <**ci**>x</**ci**> </**bvar**> <**lowlimit**> <**cn**>0</**cn**> </**lowlimit**> <**uplimit**> <**ci**>a</**ci**> </**uplimit**> <**apply**><**ci**>f</**ci**> <**ci**>x</**ci**> </**apply**> </**apply**>	<**apply**> <**int**/> <**bvar**> <**ci**>x</**ci**> </**bvar**> <**condition**> <**apply**><**in**/> <**ci**>x</**ci**> <**ci**>D</**ci**> </**apply**> </**condition**> <**apply**><**ci**>f</**ci**> <**ci**>x</**ci**> </**apply**> </**apply**>	<**apply**> <**int**/> <**bvar**> <**ci**>x</**ci**> </**bvar**> <**domainofapplication**> <**ci**>D</**ci**> </**domainofapplication**> <**apply**><**ci**>f</**ci**> <**ci**>x</**ci**> </**apply**> </**apply**>

OpenMath can model usages (a) and (c) easily enough, via its `defint` operator: in fact usage (a) is modeled on the lines of (c), as $\int_{[0,a]} f(x)dx$, which means that we need to give an eccentric[9] meaning to 'backwards' intervals in order to encode the traditional mathematical statement

$$\int_a^b f(x)dx = -\int_b^a f(x)dx. \tag{1}$$

A more logical view is to regard the two notations as different, and define ${}_{\epsilon\delta}\int_{[a,b]}$ (via limits of Riemann sums, or whatever other definition is appropriate), and then

$${}_{\epsilon\delta}\int_a^b f = \begin{cases} {}_{\epsilon\delta}\int_{[a,b]} f & a \le b \\ -{}_{\epsilon\delta}\int_{[b,a]} f & a > b \end{cases}, \tag{2}$$

whereas

$${}_{\mathrm{DA}}\int_a^b f = \left({}_{\mathrm{DA}}\int f\right)(b) - \left({}_{\mathrm{DA}}\int f\right)(a) \tag{3}$$

by definition.

Usage (b) might not worry us too much at first, since it is apparently only a variant of (c). The challenge comes when we move to multidimensional integration (in the ${}_{\epsilon\delta}\int$ sense). [BE95, p. 189] has a real integral over a curve in the complex plane,

$$\frac{1}{2\pi}\int_{|t|=R}\left|\frac{f(t)}{t^{n+1}}\right||dt| \tag{4}$$

whereas [Apo67, p. 413, exercise 4, slightly reformulated] has an integral where we clearly want to connect the variables in the integrand to the variables defining the set:

$$\int\int\int_{\left\{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\le 1\right\}}\left(\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\right)dxdydz \tag{5}$$

[9] Along the lines of "the set $[b,a]$ is the same as $[a,b]$ except that, where it appears as a range of integration, we should negate the value of the integral"! [Koh08]. It is possible to regard 'backwards integration' as an "idiom" in the sense of [LC99] and (1) as the explanation of that idiom, but this seems circular.

5 A Radical Proposal: Enhanced Binding Operators

The multiple points of view in the $\epsilon\delta$ vs. DA discussion can be seen in other situations, as witnessed by the difference between the OpenMath and MathML representations of the set $\{x|y < 1\}$ above. There seem to be two styles of thinking about mathematical objects. The first one — we will call it the **first-order style** — manifests itself as the $\epsilon\delta$-style in calculus. This style avoids passing around functions and sets as arguments to operators and uses expressions with bound variables instead. The second style — which we will call the **higher-order style** — allows functions and sets as arguments and relies heavily on this feature for conceptual clarity. It can be argued that the higher-order style is more modern[10], but arguably the first-order style still permeates much of mathematical practice. And if we take the use of mathematics in the Sciences and Engineering into account probably accounts for the vast majority of mathematical communication. Therefore we argue that both representational styles must be supported by MathML and OpenMath (and strict content MathML)

Examples like (4) and (5) show that the binding objects in OpenMath are too weak representationally to accomodate the first-order style of representation faithfully, and so force the reader into a higher-order style: we want the triple integration operator in (5) to range over a restricted domain of integration, and we want to give this domain as an *expression over the integration variables*[11], at least in $\epsilon\delta$ variant of integration. Moreover, given the discussion in Section 3 we need these variables to participate in α-conversion. How might we encode this in OpenMath? Figure 2 shows 4 alternatives[12]:

1. **In the binder** We can interpret $\int\int\int_{\left\{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leq 1\right\}}$ as a complex binding operator, as in `forallrestricted` and try to use that in a binding object. But this runs foul of the OpenMath2 dictum that the binding operator is not subject to α-conversion by its own variables; so this avenue is closed.
2. **In the body** On the other hand we can interpret the domain restriction as part of the binding object, and represent (5) as (2) in Figure 2. But this is impossible in OpenMath2, since only one OpenMath object after the `OMBVAR` element is allowed.

[10] It has gained traction in the second half of the 20^{th} century with the advent of category theory in Math and type theories in Logic

[11] Note that the objection that the original formulation in [Apo67], which was "$\int\int\int_S \ldots$ where $S = \{\cdots\}$", transcends the scope of both MathML and OpenMath, which restrict themselves to mathematical formulae. In fact MathML2 had limited support for inter-formula effects with the `declare` element, but deprecates this element in MathML3 since it cannot be defined on an intra-formula level. Thus the (important) issue of connecting bindings between different formula must be relegated to representation formats that transcend individual formulae, such as the OMDoc format [Koh06].

[12] We use boxed formulae as placeholders for their (straightforward but lengthy) OpenMath2 encodings.

3. **In the body (2)** We can solve this problem by inventing a mathematically meaningless "gluing" operator
4. **separately** It is possible to represent an integration formula in OpenMath2 that is supposedly equivalent mathematically to (5) using the Differential Algebra approach: but this is, from the $\epsilon\delta$ point of view, totally unnatural, since it is α-equivalent to the expression on the right which is unreadable for a human, and destroys commonality of formulae.

1.	<**OMBIND**> <**OMA**> <**OMS** cd="calculus_new" name="tripleintcond"/> $\boxed{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leq 1}$ </**OMA**> <**OMBVAR**>$\boxed{x, y, z}$</**OMBVAR**> $\boxed{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}}$ </**OMBIND**>	2.	<**OMBIND**> <**OMS** cd="calculus_new" name="tripleintcond"/> <**OMBVAR**>$\boxed{x, y, z}$</**OMBVAR**> $\boxed{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leq 1}$ $\boxed{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}}$ </**OMBIND**>
3.	<**OMBIND**> <**OMS** cd="calculus_new" name="tripleintcond"/> <**OMBVAR**>$\boxed{x, y, z}$</**OMBVAR**> <**OMA**> <**OMS** cd="calculus_new" name="tripleint_inner"/> $\boxed{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leq 1}$ $\boxed{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}}$ </**OMA**> </**OMBIND**>	4.	<**OMA**> <**OMS** cd="calculus_new" name="tripleintcond"/> $\boxed{\lambda x, y, z. \frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leq 1}$ $\boxed{\lambda r, s, t. \frac{r^2}{a^2}+\frac{s^2}{b^2}+\frac{t^2}{c^2}}$ </**OMA**>

Fig. 2. The Alternatives

5.	<**OMA**> <**OMS** cd="calculus_new" name="tripleintcond"/> $\boxed{\lambda x, y, z. \frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\leq 1}$ $\boxed{\lambda z, y, x. \frac{z^2}{a^2}+\frac{y^2}{b^2}+\frac{x^2}{c^2}}$ </**OMA**>

Solution 1 makes bound variables have an unusual, to say the least, scope, and solution 4 is higher-order style, so we are left with the other two. The two have quite a lot in common, since they both achieve the fundamental goal of making both the region and the integrand subject to the *same* binding operation. We can summarise the points as follows.

2: *pro*: Mathematically elegant; fits into both the XML and binary encodings of OpenMath.
2: *con*: Requires a change to the abstract description of the OpenMath standard.
3: *pro*: No change to the OpenMath standard.
3: *con*: Needs a new, mathematically meaningless, symbol such as `tripleint_inner` for each symbol such as `tripleintcond`.

Option 2 is our preferred route, and the rest of this paper assumes that, but the changes to adopt option 3 should be obvious. The changes to the OpenMath standard to adopt option 2 are in the Appendix of the full paper [DK09c].

6 Conditions in MathML

Our proposal above still leaves us with the problem to figure out the meaning of the `condition` from the examples and to specify their meaning in terms of OpenMath3 objects. MathML 2 introduces 23 examples of its usage, described in Table 1 of [DK09c], and a further 31 in Appendix C, described in Table 2 of [DK09c]. These can be roughly categorised as follows (where $a + b$ means "a in Chapter 4 and b in Appendix C").

5+14 are used to encode $\exists n \in \mathbb{N}$ or $\forall n \in \mathbb{N}$ (or equivalents). Strictly speaking, these usages are not necessary, because of the equivalences below.

$$\exists v \in S \quad p(v) \Leftrightarrow \exists v \quad (v \in S) \wedge p(v) \tag{6}$$

$$\forall v \in S \quad p(v) \Leftrightarrow \forall v \quad (v \in S) \Rightarrow p(v) \tag{7}$$

However, in practice, it would be better to have a convenient shorthand for these, hence the proposal in [DK09b] for OpenMath symbols `existsrestricted` and `forallrestricted`, which are constructors for complex binding operators that include the restricting domain of quantification.

6+4 can be replaced by the OpenMath `suchthat` construct: See section 10.1 of [DK09c].

2+2 are solved by the use of `map` in OpenMath.

Pragmatic MathML	Strict MathML
<**apply**> W <**bvar**>X</**bvar**> <**condition**>Y</**condition**> Z <**apply**>	<bind> W' <**bvar**>X</**bvar**> Z Y <bind>

Fig. 3. Translating MathML with `condition`

So we see that for all concrete operators, we have a natural strict content MathML/OpenMath equivalent. In the other cases we use the translation in Figure 3 afforded by OpenMath/strict MathML extended according to our proposal. Here W is a binding operator and X stands for any number of variables in the `bvar` construct and Y, Z are arbitrary MathML expressions. Since we have treated all concrete operators, W must be either a `ci`, `cn`, a complex MathML expression, or a `csymbol` element. We conjecture that the first two cases have not been used, since there is no plausible way to give them meaning; we propose to deprecate such usages in MathML3. In contrast to that, the `csymbol` case is an eminently legitimate use, and therefore have to provide a W' in the rule above. But in MathML2, a `csymbol` element only has a discernible meaning, if it carries a `definitionURL` attribute that points to a description D of the symbols' meaning, which will specify the meaning of the expression in terms of X, Y and Z. This description can be counted as (or turned into) a CD D' that declares a binary binding operator that can be referenced by a `csymbol` element W' which points to this declaration. Note that if D described a usage of the operator W without a `condition` qualifier, then D' must also declare the unary binding operator W; this must be different from W', since we assume OpenMath operators to have fixed arities. Finally, note that the case where W is a complex expression is analogous to the previous cases depending on the head symbol of W.

7 Lifting Associative Operators

Binary associative operators have notational peculiarities of their own. While we tend to write then as binary, as "$a+b+c$", we recognise that this is "really" one addition of three numbers, and both MathML-Content and OpenMath would represent this as a `plus` with three arguments. Mathematica distinguishes such operators as `Flat` and OpenMath's Simple Type System [Dav00] as `nassoc`. It therefore makes sense to think of applying them to collections of arguments, and mathematical notation does this all the time (see table 4).

"small"	$a_1+a_2+a_3$	$a_1a_2a_3$	$a_1\cap a_2\cap a_3$	$a_1\cup a_2\cup a_3$	$a_1\otimes a_2\otimes a_3$	$a_1\vee a_2\vee a_3$
small Unicode			225C	225B	220A	225F
"big"	$\sum_{i=1}^3 a_i$	$\prod_{i=1}^3 a_i$	$\bigcap_{i=1}^3 a_i$	$\bigcup_{i=1}^3 a_i$	$\bigotimes_{i=1}^3 a_i$	$\bigvee_{i=1}^3 a_i$
big Unicode	1350	1351	1354	1353	134E	1357

Fig. 4. "Big" operators

With the exception of $\sum$ and $\prod$, which [Dav08] regarded as being among the "irregular verbs" of mathematical notation, we can see a familiar pattern: the operator that applies to a collection of argument is "bigger" than its infix binary equivalent. The designers of Unicode[13] have done as well as might be hoped for in mapping these symbols to 'related' code points in Unicode space for the corresponding glyphs.

How are these "big" operators going to be represented? For those it "knows" about [Con03, 4.2.3.2] (the list is, with our decorations. given in Figure 5: the ones marked ¶ are no longer n-ary in *strict* MathML3),

`plus`, `times`, `max`*, `min`*, `gcd`*, `lcm`*, `mean`‡, `sdev`‡, `variance`‡, `median`‡, `mode`‡, `and`*, `or`*, `xor`†, `union`*, `intersect`*, `cartesianproduct`†, `compose`†, `eq`¶, `leq`¶, `lt`¶, `geq`¶, `gt`¶

Fig. 5. MathML 2's n-ary operators

MathML can use bound variables and conditions, so the last item from Figure 4 would be shown on the left in Figure 6. It is not clear from [Con03] whether the same construct can be applied to a user-defined operator, but it would certainly be reasonable. OpenMath, on the other hand, has an explicit lifting operator `apply_to_list`, see Figure 6 on the right.

[13] One might object that the designer of TeX did not do as well, since the last column of Figure 4 is coded using `lor` for the first row, but `bigvee` for the third. However, `lor` is in fact merely an alias for `vee`.

<**apply**>
  <**or**/>
 <**bvar**><**ci**>i</**ci**></**bvar**>
  <**lowlimit**><**cn**>1</**cn**></**lowlimit**>
  <**uplimit**><**cn**>3</**cn**></**uplimit**>
  $\boxed{a_i}$
</**apply**>
	<**OMA**>
	  <**OMS** name="apply_to_list" cd="fns2"/>
	  <**OMS** name="or" cd="logic1"/>
	  <**OMA**>
	    <**OMS** cd="list1" name="make_list"/>
	   <**OMI**>1</**OMI**>
	    <**OMI**>3</**OMI**>
	     $\boxed{\lambda i.a_i}$
	 </**OMA**>
	</**OMA**>

Fig. 6. $\bigvee$ in OpenMath and MathML

Many of the operators $\oplus$ listed in Figure 5, those we have marked *, have two additional properties:

idempotence $\forall f\ f \oplus f = f$;
monotonicity There is some discrete order $\succ$ such that $\forall f, g\ f \oplus g \succ g$.

The first means that it make sense to apply $\oplus$ to a *set*, i.e. $\bigoplus S$. The second means that it makes sense to talk about $\bigoplus_{i=1}^{\infty} s_i$, as being the point where the construct stabilises under $\succ$, or some kind of infinite object otherwise. OpenMath's construction has no problem with, say, $\bigvee F$, but MathML has to write this as $\bigvee_{p \in F} p$ and use `condition` to represent the $p \in F$.

The statistical operators (marked ‡), when applied to discrete sets, and those marked † only make sense over finite collections, but $\sum$ and $\prod$, as well as being lexically irregular in not being the infix operators writ large, are different in that they *can* have a calculus connotation. Here neither OpenMath nor MathML 3 make any clear distinctions, nor, in their defence, do the vast majority of mathematics texts. Is that sum meant to be absolutely convergent or only conditionally convergent? Only a careful analysis of the surrounding text will show, if then.

To help those authors who wish to make such distinctions, OpenMath probably ought to have a CD of symbols with finer distinctions, just as it should for the various kinds of integrals such as Cauchy Principal Value, but this is not an OpenMath/MathML issue.

8 Conclusion

We have listed four areas where MathML (1–2) and OpenMath have taken different routes to the expressivity of mathematical meaning.

In the case of MathML's `condition`, we have seen one very general concept that does not have a single formalisation, and this led to the pragmatic/strict distinction in MathML3. We have seen the utility of "restricted" quantifiers, even though they are not logically necessary, and [DK09b] proposes their addition to OpenMath.

In the case of the calculus operations, this reflected a genuine split in the approaches to the calculus operations, whether one viewed them as algebraic or analytic operations. Since neither is 'wrong', but the two *are* different (for example the "Fundamental Theorem of Calculus" is a theorem from the analytic point of view, but a definition in the algebraic view), a converged view at MathML/OpenMath 3 should incorporate both.

Acknowledgements

The unification effort described here has benefited from the input of many people, notably Olga Caprotti, David Carlisle, Sam Dooley, Christoph Lange, Paul Libbrecht, Bruce Miller, Robert Miner, Florian Rabe, Chris Rowley. The authors are indebted to David Carlisle for comments on an earlier version of the paper.

References

[Apo67] T.M. Apostol. Calculus, Volume II, 2nd edition. *Blaisdell*, 1967.

[Bar84] H.P. Barendregt. The Lambda Calculus: Its Syntax and Semantics. *North-Holland*, 1984.

[BE95] P. Borwein and T. Erdélyi. Polynomials and Polynomial Inequalities. *Springer Graduate Texts in Mathematics 161*, 1995.

[CDD+01] David Carlisle, James Davenport, Mike Dewar, N. Hur, and William Naylor. Conversion between MathML and OpenMath. Technical report, The OpenMath Society, 2001.

[Con03] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 2.0 (Second Edition): W3C Recommendation 21 October 2003. http://www.w3.org/TR/MathML2/, 2003.

[Dav00] J.H. Davenport. A Small OpenMath Type System. *ACM SIGSAM Bulletin 2*, 34:16–21, 2000.

[Dav08] J.H. Davenport. OpenMath in a (Semantic) Web. http://www.jem-thematic.net/file_private/Barcelona.pdf, 2008.

[DK09a] J.H. Davenport and M. Kohlhase. Calculus in OpenMath. *Submitted to 9th OpenMath Workshop*, 2009.

[DK09b] J.H. Davenport and M. Kohlhase. Quantifiers in OpenMath. *Submitted to 9th OpenMath Workshop*, 2009.

[DK09c] J.H. Davenport and M. Kohlhase. Unifying Math Ontologies: A tale of two standards (full paper). http://opus.bath.ac.uk/13079, 2009.

[HS08] J.R. Hindley and J.P. Seldin. Lambda-Calculus and Combinators. *Cambridge University Press*, 2008.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

[Koh08] M. Kohlhase. OpenMath3 without conditions: A Proposal for a MathML3/OM3 Calculus Content Dictionary. https://svn.openmath.org/OpenMath3/doc/blue/noconds/note.pdf, 2008.

[Lan09] Christoph Lange. OpenMath wiki. http://wiki.openmath.org, 2009.

[LC99] Z. Luo and P. Callaghan. Mathematical Vernacular and Conceptual Well-Formedness in Mathematical Language. In *Proceedings Logical Aspects of Computational Linguistics 1997*, pages 231–250, 1999.

[LGP08] Christoph Lange and Alberto González Palomo. Easily editing and browsing complex OpenMath markup with SWiM. In Paul Libbrecht, editor, *Mathematical User Interfaces Workshop 2008*, 2008.