Conditions in MathML

James Davenport
J.H.Davenport@bath.ac.uk

November 3, 2008

Abstract

JHD's attempt to look at the uses of `condition` in the MathML2 specification. Chapter 4 and Appendix C are now complete, in the sense that every use of `condition` in them has been analysed. The conclusion is that almost all of the can be represented in terms of current OpenMath, or (better but not strictly necessary) current OpenMath with two additional symbols: `forallrestricted` and `existsrestricted`.

1 Introduction

This note follows from the OpenMath tele-conference at 15:00 GMT[1] on 10/10/ /2008. See Michael Kohlhase (`m.kohlhase@jacobs-university.de`)'s mail `48F05077.7010004@jacobs-university.de`.

The summary reads as follows.

> James was tasked to make sense of the integration/differentiation examples from the MathML2 spec and make concrete suggestions for the expression-based calculus CD.

The detailed record shows that it should be a wider look at *all* uses of condition, and this version is an attempt to look at all occurrences of `condition` in the MathML2 specification.

From table 1 (page 48) we see that the first dubious case in the body of the standard is described in our section 2.3.1, where the alternative in MK's note [3] seems not to conform to OpenMath in practice. This example also appears in sections 2.8 and 2.10. From table 2 (page 49) we see that the only dubious case in Appendix C is in our section 3.3.2, which is essentially the same example.

Our section 2.3.3 illustrates a problem with multivariate definite integrals that OpenMath cannot represent directly, and that MathML requires a non-intuitive correspondence of order of variables to express.

We note that our section 2.4.2 is an excellent tribute to the power of the proposed new symbols `forallrestricted` and `existsrestricted`. JHD has been trying to find the origin of these symbols, but it seems not to be in his

[1] 16:00 BST, 17:00 CET.

archive. Memory is that MK suggested `forallrestricted`. The point is that the head of an `OMBIND` need for be a symbol, but can be a compound expression, so using

```
<OMA>
  <OMS name="forallrestricted" cd="quant2"/>
  <OMV name="S"/>
</OMA>
```

as the head is the same as using `<OMS name="forall" cd="quant1"/>` except that the variable(s) bound are restricted to range over S. Note that the bound variables do *not* explicitly appear in the restriction, so this does not fall foul of OpenMath's requirement highlighed on page 31. We should note what `forallrestricted` does, and does not, encode.

does $\forall n \in \mathbf{N}$, $\forall m, n \in \mathbf{N}$, $\forall x \in [0, 1]$, $\forall x \in (0, \infty)$ (but not the equivalent $\forall x > 0$).

does not $\forall n \in \mathbf{N}, x \in \mathbf{R}$ (this needs two nested `forallrestricted`s), $\forall n > 2$, $\forall m < n \in \mathbf{N}$.

It is a legitimate argument that this `forallrestricted` symbol is privileging the use of $\in$ in what MathML called `condition`s.

2 Chapter 4

This chapter is the substantive specification of MathML (Content).

2.1 4.2.1.8 The use of qualifier elements

This section introduces the `condition` element, in what might be called a "proof by example" style.

2.1.1 4.2.1.8(1) The use of qualifier elements

This section contains the following quotation.

> A `condition` element can be used to place restrictions directly on the bound variable. This allows MathML to define sets by rule, rather than enumeration. 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>
  <ci> x </ci>
</set>
```

This can be converted into OpenMath by means of `suchthat`: here is an OpenMath example from `set1` reworked to encode the same mathematics.

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
       cdbase="http://www.openmath.org/cd">
  <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"/>
        <OMV name="x"/>
        <OMI> 1 </OMI>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

We note that the OpenMath makes it explicit that it is $(-\infty, 1)$ that is meant, not, say, $[0, 1)$, and equally that it is $\mathbf{R}$, not $\mathbf{Z}$ or some other set, that is the base type.

2.1.2 4.2.1.8(2) The use of qualifier elements

This section contains the following quotation.

> Another typical use is the "lifting" of n-ary operators to "big operators", for instance the n-ary `union` operator to the `union` operator over sets, as the union of the U-complements over a family F of sets in this construction.

```
<apply>
  <union/>
  <bvar><ci>S</ci></bvar>
  <condition>
    <apply><in/><ci>S</ci><ci>F</ci></apply>
  </condition>
  <apply><setdiff/><ci>U</ci><ci>S</ci></apply>
</apply>
```

This falls foul of the ambiguity in MathML's `union` constructor[2] highlighted in [1, especially slide 14]. The best OpenMath translation would seem to be on the following lines.

```
<OMA>
  <OMS name="big_union" cd="set3"/>
  <OMA>
    <OMS name="map" cd="set1"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR> <OMV name="S"/> </OMBVAR>
      <OMA>
        <OMS name="setdiff" cd="set1"/>
        <OMV name="U"/>
        <OMV name="S"/>
      </OMA>
    </OMBIND>
    <OMV name="F"/>
  </OMA>
</OMA>
```

2.2 4.2.2.2 Constructors

This contains the sentence

> For example, a `bvar` and a `condition` element can be used to define lists where membership depends on satisfying certain conditions.

No example is given here, but the example in 4.2.1.8 (our section 2.1.1) could be regarded as typical.

2.3 4.2.3.2 Operators taking Qualifiers

This section lists `condition` among the qualifiers, and the operators taking qualifiers (not necessarily `condition`) as follows (* indicates that section 4.2.3 of the MathML specification states that they do not take `condition` as a qualifier).

[2] At the end of section 4.2.3 of the MathML specification, we read

> If qualifiers are used, they should be followed by a single child element representing a function or an expression in the bound variables specified in the `bvar` qualifiers. Mathematically the operation is then taken to be over the arguments generated by this function ranging over the specified domain of application, rather than over an explicit list of arguments as is the case when qualifier schemata are not used.

A purist might objection that the presence of the qualifier is changing the fundamental semantics of the n-ary operator.

operators `int`[3], `sum`[4], `product`[5], `root, diff`*, `partialdiff`*, `limit`[6], `log`*, `moment`*, `forall`[7], `exists`.

n-ary operators `plus, times, max`[8], `min`[9], `gcd, lcm, mean, sdev, variance, median, mode, and, or, xor, union`[10], `intersect, cartesianproduct, compose, eq, leq, lt, geq, gt`.

user defined operators `csymbol, ci`.

missing (or not regarded as operators in MathML) `set`[11], `list`[12], `interval`[13] `matrix`[14].

spuriously absent `subset`[15] and in theory all other n-ary relations — see section 2.13.

> The (`lowlimit`,`uplimit`) pair, the `interval` and the `condition` are all shorthand notations specifying a particular domain of application and should not be used if `domainofapplication` is used.

It is not clear to the current author how the example in section 3.1 can be cast in this mould, though.

2.3.1 4.2.3.2 Operators taking Qualifiers (1)

`condition` has the following example.

```
<apply>
  <int/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><in/><ci>x</ci><ci type="set">C</ci></apply>
  </condition>
  <apply><sin/><ci>x</ci></apply>
</apply>
```

This is a special case of the example discussed in section 3.3.2

[3]Sections 2.3.1, 3.3.2, 3.5, 3.17.
[4]Sections 2.14 and 3.23.
[5]Section 3.24.
[6]Sections 2.16, 3.19, 3.25, 3.29.2.
[7]Sections 3.2, 3.4, 3.6, 3.7, 3.8, 3.11, 3.12, 3.14, 3.15, 3.16. 3.18, 3.27, 3.28, 3.29.1.
[8]Sections 2.5.3, 3.9.
[9]Sections 2.6.1, 3.10.
[10]Section 2.1.2.
[11]Sections 2.1.1, 3.20.
[12]Section 3.21.
[13]Section 3.1
[14]Section 3.26.
[15]Sections 2.13 and 3.22 — note that we can make no sense of this last example.

2.3.2 4.2.3.2 Operators taking Qualifiers (2)

It is also stated that

```
<apply>
  <int/>
  <bvar><ci>x</ci></bvar>
  <lowlimit><cn>0</cn></lowlimit>
  <uplimit><cn>1</cn></uplimit>
  <apply><power/><ci>x</ci><cn>2</cn></apply>
</apply>
```

(whose OpenMath equivalent in terms of `calculus1` is

```
<OMA>
  <OMS cd="calculus1" name="defint"/>
  <OMA>
    <OMS name="interval_cc" cd="interval1"/>
    <OMI> 0 </OMI>
    <OMI> 1 </OMI>
  </OMA>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR> <OMV name="x"/> </OMBVAR>
    <OMA>
      <OMS cd="arith1" name="power"/>
      <OMV name="x"/>
      <OMI> 2 </OMI>
    </OMA>
  </OMBIND>
</OMA>
```

) can be written as

```
<apply>
  <int/>
  <bvar><ci>x</ci></bvar>
  <domainofapplication>
    <set>
      <bvar><ci>t</ci></bvar>
      <condition>
        <apply>
          <and/>
          <apply><leq/><cn>0</cn><ci>t</ci></apply>
          <apply><leq/><ci>t</ci><cn>1</cn></apply>
        </apply>
      </condition>
      <ci>t</ci>
```

```
    </set>
  </domainofapplication>
  <apply><power/><ci>x</ci><cn>2</cn></apply>
</apply>
```

The OpenMath equivalent of this would probably be the following.

```
<OMA>
  <OMS cd="calculus1" name="defint"/>
  <OMA>
    <OMS cd="set1" name="suchthat"/>
    <OMS name="R" cd="setname1"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR> <OMV name="t"/> </OMBVAR>
      <OMA>
        <OMS name="and" cd="logic1"/>
        <OMA>
          <OMS cd="relation1" name="gt"/>
          <OMV name="t"/>
          <OMI> 0 </OMI>
        </OMA>
        <OMA>
          <OMS cd="relation1" name="lt"/>
          <OMV name="t"/>
          <OMI> 1 </OMI>
        </OMA>
      </OMA>
    </OMBIND>
  </OMA>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR> <OMV name="x"/> </OMBVAR>
    <OMA>
      <OMS cd="arith1" name="power"/>
      <OMV name="x"/>
      <OMI> 2 </OMI>
    </OMA>
  </OMBIND>
</OMA>
```

2.3.3 4.2.3.2 Operators taking Qualifiers (3)

The example continues as follows.

> This use extends to multivariate domains by using extra bound variables and a domain corresponding to a cartesian product as in:

```
<apply>
  <int/>
  <bvar><ci>x</ci></bvar>
  <bvar><ci>y</ci></bvar>
  <domainofapplication>
    <set>
      <bvar><ci>t</ci></bvar>
      <bvar><ci>u</ci></bvar>
      <condition>
        <apply>
          <and/>
          <apply><leq/><cn>0</cn><ci>t</ci></apply>
          <apply><leq/><ci>t</ci><cn>1</cn></apply>
          <apply><leq/><cn>0</cn><ci>u</ci></apply>
          <apply><leq/><ci>u</ci><cn>1</cn></apply>
        </apply>
      </condition>
      <list><ci>t</ci><ci>u</ci></list>
    </set>
  </domainofapplication>
  <apply>
    <times/>
    <apply><power/><ci>x</ci><cn>2</cn></apply>
    <apply><power/><ci>y</ci><cn>3</cn></apply>
  </apply>
</apply>
```

> Note that the order of bound variables of the integral must correspond to the order in the `list` used by the `set` constructor in the `domainofapplication`.

OpenMath 2 as it (and its CDs) exists has no immediate answer to this, since `defint` from `calculus1` explicitly only integrates unary functions. Obviously one could replace it by two nested unary integrations. Trying to represent it directly would fall foul of the potential ambiguity referred to in the quotation immediately above. It seems to the author that one really wants some way of representing the following expression:

$$\int_{x=0}^{1}\int_{y=0}^{1} x^2y^3\mathrm{d}x\mathrm{d}y, \tag{1}$$

i.e. explicitly linking the variables to the bounds.

2.3.4 4.2.3.2 Operators taking Qualifiers (4)

The text on `forall` gives the following example.

```
<apply>
  <forall/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/>
      <ci> x </ci><cn> 9 </cn>
    </apply>
  </condition>
  <apply><lt/>
    <ci> x </ci><cn> 10 </cn>
  </apply>
</apply>
```

This could be solved with the forall/implies encoding, as in

```
<OMBIND>
  <OMS name="forall" cd="quant1"/>
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMA>
    <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="lt" cd="relation1"/>
      <OMV name="x"/>
      <OMI> 9 </OMI>
    </OMA>
    <OMA>
      <OMS name="lt" cd="relation1"/>
      <OMV name="x"/>
      <OMI> 10 </OMI>
    </OMA>
  </OMA>
</OMBIND>
```

Alternatively, we could use a new symbol `forallrestricted`, as in the following.

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMA>
      <OMS name="suchthat" cd="set1"/>
      <OMS cd="setname1" name="R"/>
      <OMBIND>
        <OMS cd="fns1" name="lambda"/>
        <OMBVAR> <OMV name="x"/> </OMBVAR>
        <OMA>
          <OMS name="lt" cd="relation1"/>
```

```
          <OMV name="x"/>
          <OMI> 9 </OMI>
        </OMA>
      </OMBIND>
    </OMA>
  </OMA>
  <OMA>
    <OMS name="lt" cd="relation1"/>
    <OMV name="x"/>
    <OMI> 10 </OMI>
  </OMA>
</OMBIND>
```

2.4 4.2.5 Conditions

2.4.1 First example

```
<apply>
  <exists/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/>
      <apply>
        <power/>
        <ci>x</ci>
        <cn>5</cn>
      </apply>
      <cn>3</cn>
    </apply>
    </condition>
    <true/>
</apply>
```

The author's first reaction was "why a condition: isn't this MathML equivalent (and shorter)?"

```
<apply>
  <exists/>
  <bvar><ci> x </ci></bvar>
  <apply><lt/>
    <apply>
      <power/>
      <ci>x</ci>
      <cn>5</cn>
    </apply>
    <cn>3</cn>
  </apply>
```

```
</apply>
```

Certainly this would be the obvious OpenMath encoding.

```
<OMBIND>
  <OMS name="exists" cd="quant1"/>
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMA>
    <OMS name="lt" cd="relation1"/>
    <OMA>
      <OMS name="power" cd="arith1"/>
      <OMV name="x"/>
      <OMI> 5 </OMI>
    </OMA>
    <OMI> 3 </OMI>
  </OMA>
</OMBIND>
```

2.4.2 Second example

```
<apply>
  <forall/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><in/>
      <ci>x</ci>
        <csymbol encoding="OpenMath"
          definitionURL="http://www.openmath.org/cd/setname1#N">
          N
        </csymbol>
    </apply>
  </condition>

  <apply><exists/>
     <bvar><ci>p</ci></bvar>
     <bvar><ci>q</ci></bvar>
     <condition>
       <apply><and/>
         <apply><in/><ci>p</ci>
           <csymbol encoding="OpenMath"
             definitionURL="http://www.openmath.org/cd/setname1#P">
             P
           </csymbol>
         </apply>
         <apply><in/><ci>q</ci>
           <csymbol encoding="OpenMath"
             definitionURL="http://www.openmath.org/cd/setname1#P">
```

```
            P
           </csymbol>
         </apply>
       </apply>
     </condition>
     <apply><eq/>
        <apply><plus/><ci>p</ci><ci>q</ci></apply>
        <apply><times/><cn>2</cn><ci>x</ci></apply>
     </apply>
   </apply>
</apply>
```

While this would succumb to the forall/implies and exists/and encodings, it is a better tribute to the power of `forallrestricted` etc.

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMS name="N" cd="setname1"/>
  </OMA>
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMBIND>
    <OMA>
      <OMS name="existsrestricted" cd="quant2"/>
      <OMS name="P" cd="setname1"/>
    </OMA>
    <OMBVAR> <OMV name="p"/> <OMV name="q"/> </OMBVAR>
    <OMA>
      <OMS name="eq" cd="relation1"/>
      <OMA>
        <OMS name="times" cd="arith11"/>
        <OMI> 2 </OMI>
        <OMV name="x"/>
      </OMA>
      <OMA>
        <OMS name="plus" cd="arith11"/>
        <OMV name="p"/>
        <OMV name="q"/>
      </OMA>
    </OMA>
  </OMBIND>
</OMBIND>
```

2.4.3 Third example

```
<apply>
  <exists/>
```

```
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/><ci>x</ci><cn>3</cn></apply>
  </condition>
  <apply>
    <eq/>
    <apply>
      <power/><ci>x</ci><cn>2</cn>
    </apply>
    <cn>4</cn>
  </apply>
</apply>
```

This works well with `existsrestricted`.

```
<OMBIND>
  <OMA>
    <OMS name="existsrestricted" cd="quant2"/>
    <OMA>
      <OMS name="suchthat" cd="set1"/>
      <OMS cd="setname1" name="R"/>
      <OMBIND>
        <OMS cd="fns1" name="lambda"/>
        <OMBVAR> <OMV name="x"/> </OMBVAR>
        <OMA>
          <OMS name="lt" cd="relation1"/>
          <OMV name="x"/>
          <OMI> 3 </OMI>
        </OMA>
      <OMBIND>
    </OMA>
  </OMA>
  <OMA>
    <OMS name="eq" cd="relation1"/>
    <OMA>
      <OMS name="power" cd="arith1"/>
      <OMV name="x"/>
      <OMI> 2 </OMI>
    </OMA>
    <OMI> 4 </OMI>
  </OMA>
</OMBIND>
```

2.5 4.4.2.7.2 Examples (of condition)

2.5.1 First example

```
<condition>
  <apply><in/><ci> x </ci><ci type="set"> A </ci></apply>
</condition>
```

As a free-standing piece of code, it is hard to see what this means.

2.5.2 Second example

```
<condition>
  <apply>
    <and/>
    <apply><gt/><ci> x </ci><cn> 0 </cn></apply>
    <apply><lt/><ci> x </ci><cn> 1 </cn></apply>
  </apply>
</condition>
```

As a free-standing piece of code, it is hard to see what this means, but it crops up as a fragment of the next example.

2.5.3 Third example

```
<apply>
  <max/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply> <and/>
      <apply><gt/><ci> x </ci><cn> 0 </cn></apply>
      <apply><lt/><ci> x </ci><cn> 1 </cn></apply>
    </apply>
  </condition>
  <apply>
    <minus/>
    <ci> x </ci>
    <apply>
      <sin/>
      <ci> x </ci>
    </apply>
  </apply>
</apply>
```

As OpenMath does not have a `max` operator acting on functions, the nearest translation would seem to be the following.

```
<OMA>
```

```
  <OMS name="max" cd="minmax1"/>
  <OMA>
    <OMS name="map" cd="set1"/>
    <OMBIND>
      <OMS name="lambda" cd="fns1"/>
      <OMBVAR> <OMV name="x"/> </OMBVAR>
      <OMA>
        <OMS name="minus" cd="arith1"/>
        <OMV name="x"/>
        <OMA>
          <OMS name="sin" cd="transc1"/>
          <OMV name="x"/>
        </OMA>
      </OMA>
    </OMBIND>
    <OMA>
      <OMS name="interval_oo" cd="interval1"/>
      <OMI> 0 </OMI>
      <OMI> 1 </OMI>
    </OMA>
  </OMA>
</OMA>
```

2.6 4.4.3.4 Maximum and minimum (max, min)

This contains the following examples.

2.6.1 4.4.3.4 max, min (1)

```
<apply>
  <min/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><notin/><ci> x </ci><ci type="set"> B </ci></apply>
  </condition>
  <apply>
      <power/>
    <ci> x </ci>
    <cn> 2 </cn>
  </apply>
</apply>
```

This is somewhat hard to interpret: if $x \notin B$, what *can* we say about x? In general terms, though, a solution such as in our section 3.9 seems appropriate.

2.6.2 4.4.3.4 max, min (2)

```
<apply>
  <max/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci>x</ci><ci type="set">B</ci></apply>
      <apply><notin/><ci>x</ci><ci type="set">C</ci></apply>
    </apply>
  </condition>
  <ci>x</ci>
</apply>
```

This is a clear case of suchthat.

```
<OMA>
  <OMS name="max" cd="minmax1"/>
  <OMA>
    <OMS name="suchthat" cd="set1"/>
    <OMV name="B"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR> <OMV name="x"/> </OMBVAR>
      <OMA>
        <OMS name="notin" cd="set1"/>
        <OMV name="x"/>
        <OMV name="C"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMA>
```

2.7 4.4.3.17 Universal quantifier (forall)

This contains the following examples.

2.7.1 4.4.3.17 forall (1)

```
<apply>
  <forall/>
  <bvar><ci> p </ci></bvar>
  <bvar><ci> q </ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci> p </ci><rationals/></apply>
      <apply><in/><ci> q </ci><rationals/></apply>
```

```
      <apply><lt/><ci> p </ci><ci> q </ci></apply>
    </apply>
  </condition>
  <apply><lt/>
      <ci> p </ci>
      <apply>
          <power/>
        <ci> q </ci>
        <cn> 2 </cn>
      </apply>
  </apply>
</apply>
```

This will clearly succumb to the forall/implies encoding. `forallrestricted` will cope with the `<apply><in/><ci> p </ci><rationals/></apply>` clauses, but not, as currently suggested, with the $p < q$ clause. This would give the following.

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMS name="Q" cd="setname1"/>
  </OMA>
  <OMBVAR> <OMV name="p"/> <OMV name="q"/> </OMBVAR>
  <OMA>
    <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="lt" cd="logic1"/>
      <OMV name="p"/>
      <OMV name="q"/>
    </OMA>
    <OMA>
      <OMS name="lt" cd="logic1"/>
      <OMV name="p"/>
      <OMA>
        <OMS name="power" cd="arith1"/>
        <OMV name="q"/>
        <OMI> 2 </OMI>
      </OMA>
    </OMA>
  </OMA>
</OMBIND>
```

2.7.2 4.4.3.17 `forall` (2)

```
<apply>
  <forall/>
  <bvar><ci> n </ci></bvar>
```

```
  <condition>
    <apply><and/>
      <apply><gt/><ci> n </ci><cn> 0 </cn></apply>
      <apply><in/><ci> n </ci><integers/></apply>
    </apply>
  </condition>
  <apply>
    <exists/>
    <bvar><ci> x </ci></bvar>
    <bvar><ci> y </ci></bvar>
    <bvar><ci> z </ci></bvar>
    <condition>
      <apply><and/>
        <apply><in/><ci> x </ci><integers/></apply>
        <apply><in/><ci> y </ci><integers/></apply>
        <apply><in/><ci> z </ci><integers/></apply>
      </apply>
    </condition>
    <apply>
      <eq/>
      <apply>
        <plus/>
        <apply><power/><ci> x </ci><ci> n </ci></apply>
        <apply><power/><ci> y </ci><ci> n </ci></apply>
      </apply>
      <apply><power/><ci> z </ci><ci> n </ci></apply>
    </apply>
  </apply>
</apply>
```

Again, the restricted quantifiers help a great deal. but not perfectly.

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMS name="Z" cd="setname1"/>
  </OMA>
  <OMBVAR> <OMV name="n"/> </OMBVAR>
  <OMA>
  <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="gt" cd="relation1"/>
      <OMV name="n"/>
      <OMI> 0 </OMI>
    </OMA>
    <OMBIND>
      <OMA>
```

```
        <OMS name="existsrestricted" cd="quant2"/>
        <OMS name="Z" cd="setname1"/>
      </OMA>
      <OMBVAR> <OMV name="x"/> <OMV name="y"/> <OMV name="z"/> </OMBVAR>
      <OMA>
        <OMS name="eq" cd="relation1"/>
        <OMA>
          <OMS name="plus" cd="arith1"/>
          <OMA>
            <OMS name="power" cd="arith1"/>
            <OMV name="x"/>
            <OMV name="n"/>
          </OMA>
          <OMA>
            <OMS name="power" cd="arith1"/>
            <OMV name="y"/>
            <OMV name="n"/>
          </OMA>
        </OMA>
        <OMA>
          <OMS name="power" cd="arith1"/>
          <OMV name="z"/>
          <OMV name="n"/>
        </OMA>
      </OMA>
    </OMBIND>
  </OMA>
</OMBIND>
```

2.8 4.4.5.1 Integral (int)

This contains the following example.

```
<apply>
  <int/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><in/>
      <ci> x </ci>
      <ci type="set"> D </ci>
    </apply>
  </condition>
  <apply><ci type="function"> f </ci>
    <ci> x </ci>
  </apply>
</apply>
```

Up to renaming, this is identical to the example in section 3.3.2, and the same comments apply.

2.9 4.4.5.6.1 Bound variable (bvar)

This contains the following example, which is mainly meant to illustrate the use of the id= construct.

```
<set>
  <bvar><ci id="var-x"> x </ci></bvar>
  <condition>
    <apply>
      <lt/>
      <ci definitionURL="#var-x"> x </ci>
      <cn> 1 </cn>
    </apply>
  </condition>
</set>
```

From our point of view, this looks like a suchthat, except that the universe is unspecified in the MathML. Taking a guess for this, we end up with the following OpenMath.

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

XML id= tags could probably be used here.[1] This might also be an occasion for OMR. EdNote(1)

2.10 4.4.5.6.2 Bound variable (bvar)

Up to renaming, this is identical to the example in section 3.3.2, and the same comments apply.

[1] EdNote: Anyone wish to do so?

```
<apply>
  <int/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><in/><ci> x </ci><ci> D </ci></apply>
  </condition>
  <apply><ci type="function"> f </ci>
    <ci> x </ci>
  </apply>
</apply>
```

2.11 4.4.6.1.2 Set (set)

This contains the following example.

```
<set>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><and/>
      <apply><lt/>
        <ci> x </ci>
        <cn> 5 </cn>
      </apply>
      <apply><in/>
        <ci> x </ci>
        <naturalnumbers/>
      </apply>
    </apply>
  </condition>
  <ci> x </ci>
</set>
```

Again, this fits very naturally as a `suchthat`, indeed probably more naturally than the original MathML.

```
<OMA>
  <OMS cd="set1" name="suchthat"/>
  <OMS cd="setname1" name="N"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR> <OMV name="x"/> </OMBVAR>
    <OMA>
      <OMS cd="relation1" name="lt"/>
      <OMV name="x"/>
      <OMI> 5 </OMI>
    </OMA>
  </OMBIND>
</OMA>
```

2.12 4.4.6.2.2 List (`list`)

This contains the following example.

```
<list order="numeric">
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/>
      <ci> x </ci>
      <cn> 5 </cn>
    </apply>
  </condition>
  <ci> x </ci>
</list>
```

This has no direct equivalent in OpenMath, not least because there is no equivalent of `order="numeric"`, but also because one has to assume that the list is being selected from **N**, because if it were selected from **Z** or **R** is would have no infimum.

The best translation is probably the following.

```
<OMA>
  <OMS name="integer_interval" cd="interval1"/>
  <OMI> 0 </OMI>
  <OMI> 4 </OMI>
</OMA>
```

The reader may protest that `integer_interval` is "special", but surely no more so than implicitly assuming selection from **N**.

2.13 4.4.6.7 Subset (`subset`)

This contains the following sentence.

> The subset element is an n-ary set relation (see Section 4.2.4 Relations). As an n-ary operator, its operands may also be generated as described in [n-ary operators] Therefore it may take qualifiers.

This appears to justify the following example.

```
<apply>
  <subset/>
  <bvar><ci type="set">S</ci></bvar>
  <condition>
    <apply><in/>
      <ci>S</ci>
      <ci type="list">T</ci>
    </apply>
  </condition>
```

```
  <ci>S</ci>
</apply>
```

As far as the present author can see, this states that the list T is linearly ordered by $\subseteq$. OpenMath does not, and seems to have decided that it will not[16], have n-ary relations of this form.

Any translation has therefore to be loose: we propose the following[17].

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMS name="N" cd="setname1"/>
  </OMA>
  <OMBVAR> <OMV name="i"/> <OMV name="j"/> </OMBVAR>
  <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="lt" cd="relation1"/>
      <OMV name="i"/>
      <OMV name="j"/>
    <OMA>
      <OMS name="subset" cd="set1"/>
      <OMA>
        <OMV name="T"/>
        <OMV name="i"/>
      </OMA>
      <OMA>
        <OMV name="T"/>
        <OMV name="j"/>
      </OMA>
    </OMA>
    </OMA>
</OMBIND>
```

2.14 4.4.7.1 Sum (sum)

This contains the following example.

```
<apply>
  <sum/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply> <in/>
      <ci> x </ci>
```

[16] MK posed this question in e-mail of 21/9/2008, tracked as `http://wiki.openmath.org/?title=cd%3Arelation1`.

[17] Which is conditional on assuming that this is the right way to select elements from a `list` — a debate which we have had, but I cannot remember the resolution.

```
      <ci type="set"> B </ci>
    </apply>
  </condition>
  <apply><ci type="function"> f </ci>
    <ci> x </ci>
  </apply>
</apply>
```

This has an obvious translation into OpenMath, in which x isn't even needed. It could, of course, be supplied by replaced f by $\lambda x.f(x)$.

```
<OMA>
  <OMS name="sum" cd="arith1"/>
  <OMV name="B"/>
  <OMV name="f"/>
</OMA>
```

2.15 4.4.7.2 Product (product)

This contains the following example.

```
<apply>
  <product/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply> <in/>
      <ci> x </ci>
      <ci type="set"> B </ci>
    </apply>
  </condition>
  <apply><ci type="function"> f </ci>
    <ci> x </ci>
  </apply>
</apply>
```

The same remarks as in the previous section apply.

2.16 4.4.7.3 Limit (limit)

This contains the following example.

```
<apply>
  <limit/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply>
      <tendsto type="above"/>
      <ci> x </ci>
```

```
      <ci> a </ci>
    </apply>
  </condition>
  <apply><sin/>
     <ci> x </ci>
  </apply>
</apply>
```

This can be translated into OpenMath as follows.

```
<OMA>
  <OMS cd="limit1" name="limit"/>
  <OMV name="a"/>
  <OMS cd="limit1" name="above"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR> <OMV name="x"/> </OMBVAR>
    <OMA>
      <OMS name="sin" cd="transc1"/>
      <OMV name="x"/>
    </OMA>
  </OMBIND>
</OMA>
```

3 Appendix C

3.1 C.2.2.4 MMLdefinition: `interval`

This contains the following example.

```
<interval>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><lt/><cn>0</cn><ci>x</ci></apply>
    </condition>
</interval>
```

Presumably this represents $(0, \infty)$. Equally, this is presumably intended to close over `<interval>`, i.e. x is bound in this expression, and freely α-convertible. However, it seems to JHD to be purely "luck" that this defines an interval. What about the following?

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

```
          <ci>x</ci>
          <cn>2</cn>
          </apply>
      </apply>
    </condition>
</interval>
```

By the same logic, this is $(-\infty, -1) \cup (1, \infty)$. As far as JHD can see, this use of `interval` is basically a declaration of a set, coupled with an assertion that that set is in fact an interval, which assertion is, in fact, true in the first case, but not the second.

The declaration of the set can be done with `suchthat`, the assertion is a problem OpenMath has not really addressed.

It is also not clear that this fragment is in fact legal. The specification says elsewhere [4.4.2.4.1] that

> The interval element expects two child elements that evaluate to real numbers.

3.2 C.2.2.5 MMLdefinition: `inverse`

This contains the following example (our formatting).

```
<apply><forall/>
  <bvar><ci>y</ci></bvar>
  <bvar><ci type="function">f</ci></bvar>
  <condition>
    <apply><in/>
      <ci>y</ci>
      <apply>
        <csymbol definitionURL="domain">
          <mtext>Domain</mtext></csymbol>
        <apply><inverse/><ci type="function">f</ci></apply>
      </apply>
    </apply>
  </condition>
  <apply><eq/>
    <apply><ci type="function">f</ci>
      <apply><apply><inverse/><ci type="function">f</ci></apply>
        <ci>y</ci>
      </apply>
    </apply>
    <ci>y</ci>
  </apply>
</apply>
```

The associated textual description is

```
ForAll( y, such y in domain( f^(-1) ), f( f^(-1)(y) ) = y
```

which does not include the fact that f is in the scope of $\forall$.

This usage seems to be basically a "typed quantifier", and as such could look like the following.

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMA>
      <OMS name="domain" cd="fns1"/>
      <OMA>
        <OMS name="inverse" cd="fns1"/>
        <OMV name="f"/>
      </OMA>
    </OMA>
  </OMA>
  <OMBVAR> <OMV name="y"/> </OMBVAR>
  <OMA>
    <OMS name="eq" cd="relation1"/>
    <OMA>
      <OMV name= "f"/>
      <OMA>
        <OMA>
          <OMS name="inverse" cd="fns1"/>
          <OMV name= "f"/>
        </OMA>
        <OMV name= "y"/>
      </OMA>
    </OMA>
    <OMV name= "y"/>
  </OMA>
</OMBIND>
```

This resembles the "vernacular", and does not bind f. If we wanted to do so, along the lines of the MathML, we would have to wrap the whole thing in another `forall`. The two cannot be combined, as we want to be able to α-convert f in the argument of `forallrestricted`, and, as MK's note of the tele-conference reads:

> In particular, we do not want to accept the occurrence of the bound variable in the (complex) binding operator. In particular, the OpenMath2 standard restricts alpha-conversion to the second and third children of the OMBIND, which is consistent with this view.

See section 3.3.2 below for an example of where this rule bites.

3.3 C.2.2.7 MMLdefinition: `condition`

This section contains two examples.

3.3.1 C.2.2.7(1) MMLdefinition: `condition`

```
<condition>
  <apply><lt/>
    <apply><power/><ci>x</ci><cn>5</cn></apply>
    <cn>3</cn>
  </apply>
</condition>
```

It is hard to understand the meaning of this fragment in isolation (and presumably the reader was not intended to do so). If it was intended to be part of an encoding of a set (mathematically, representing the interval $(-\infty, 3^{1/5})$, or possibly $[0, 3^{1/5})$), then `suchthat` from `set1` is appropriate. Here is the example from `set1` reworked to match the MathML example (as $(-\infty, 3^{1/5})$: if one wanted $[0, 3^{1/5})$ one would have to add $x \geq 0$, or change from $\mathbf{R}$ as the base set).

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
       cdbase="http://www.openmath.org/cd">
  <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"/>
        <OMA>
          <OMS cd="arith1" name="power"/>
          <OMV name="x"/>
          <OMI> 5 </OMI>
        </OMA>
        <OMI> 3 </OMI>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

We should note that the OpenMath makes it clear that x is bound by the (`OMBIND` whose first child is the) `lambda`, whereas the MathML, being only a fragment, does not state the scope.

3.3.2 C.2.2.7(2) MMLdefinition: `condition`

(Our formatting.)

```
<apply><int/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><in/><ci>x</ci><ci type="set">C</ci></apply>
  </condition>
  <apply><ci type="function">f</ci><ci>x</ci></apply>
</apply>
```

This seems, to JHD, to be typical of the confusion that can arise over `conditions`. Let us look at this expression in the vernacular:

$$\int_{x \in C} f(x) \mathrm{d}x, \tag{2}$$

and observe that this is probably equivalent to $\int_C f(x)\mathrm{d}x$.

In terms of `calculus1` (calculus with functions [2]) this can be expressed as

```
<OMA>
  <OMS cd="calculus1" name="defint"/>
  <OMV name="C"/>
  <OMV name="f"/>
</OMA>
```

and x doesn't appear at all. If one wants it to, then one writes

```
<OMA>
  <OMS cd="calculus1" name="defint"/>
  <OMV name="C"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR>
      <OMV name="x"/>
    </OMBVAR>
    <OMA>
      <OMV name="f"/>
      <OMV name="x"/>
    </OMA>
  </OMBIND>
</OMA>
```

as in the example in `calculus1`.

MK's note [3] on `calculus3` (calculus with expressions [2]), suggests (in MathML syntax)

```
<bind>
  <apply>
    <csymbol cd="calculus3">defintset</csymbol>
    <ci>C</ci>
  </apply>
```

```
  <bvar><ci>x</ci></bvar>
  <apply><ci>f</ci><ci>x</ci></apply>
</bind>
```

which is a precise translation of the previous one from the language of functions to expressions. In OpenMath syntax it would be the following[18].

```
<OMBIND>
  <OMA>
    <OMS cd="calculus3" name="defintset"/>
    <OMV name="C"/>
  </OMA>
  <OMBVAR>
    <OMV name="x"/>
  </OMBVAR>
  <OMA>
    <OMV name="f"/>
    <OMV name="x"/>
  </OMA>
</bind>
```

MK's note also suggests an alternative way of expressing (2), which in MathML syntax would be the following.

```
<bind>
  <apply>
    <csymbol cd="calciulus3">defintcond</csymbol>
    <apply><in/>
      <ci>x</ci>
      <ci>C</ci>
    </apply>
  </apply>
  <bvar><ci>x</ci></bvar>
  <apply><ci>f</ci><ci>x</ci></apply>
</bind>
```

In OpenMath syntax it would be the following.

```
<OMBIND>
  <OMA>
    <OMS cd="calculus3" name="defintcond"/>
    <OMA>
      <OMS name="in" cd="set1"/>
      <OMV name="x"/>
      <OMV name="C"/>
    </OMA>
```

[18] In the circulated version of `calculus3`, there appear to be two symbols called `defint`. The second should presumably be `defintset`.

```
  </OMA>
  <OMBVAR>
    <OMV name="x"/>
  </OMBVAR>
  <OMA>
    <OMV name="f"/>
    <OMV name="x"/>
  </OMA>
</bind>
```

This, however, falls foul of

> the OpenMath2 standard restricts alpha-conversion to the second and third children of the OMBIND

and hence JHD does not see how `defintcond` as postulated can make its way into OpenMath.

3.4 C.2.2.14 MMLdefinition: `image`

This section contains the following example.

```
<apply><forall/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><in/>
      <ci>x</ci>
      <apply><image/><ci>f</ci></apply>
    </apply>
  </condition>
  <apply><in/>
    <ci>x</ci>
    <apply><codomain/><ci>f</ci></apply>
  </apply>
</apply>
```

The following remarks could be made.

1. This cries out for `forallrestricted`.

   ```
   <OMBIND>
     <OMA>
       <OMS name="forallrestricted" cd="quant2"/>
       <OMA>
         <OMS name="image" cd="fns1"/>
         <OMV name="f"/>
       </OMA>
     </OMA>
   ```

```
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMA>
    <OMS name="in" cd="set1"/>
    <OMV name="x"/>
    <OMA>
      <OMS name="codomain" cd="fns1"/>
      <OMV name= "f"/>
    </OMA>
  </OMA>
</OMBIND>
```

2. It is a pretty good case for the "forall with implies" trick.

```
<OMBIND>
  <OMS name="forall" cd="quant1"/>
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMA>
    <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="in" cd="set1"/>
      <OMV name="x"/>
      <OMA>
        <OMS name="image" cd="fns1"/>
        <OMV name= "f"/>
      </OMA>
    </OMA>
    <OMA>
      <OMS name="in" cd="set1"/>
      <OMV name="x"/>
      <OMA>
        <OMS name="codomain" cd="fns1"/>
        <OMV name= "f"/>
      </OMA>
    </OMA>
  </OMA>
</OMBIND>
```

3. Why use quantifiers at all?

```
<OMA>
  <OMS name="subset" cd="set1"/>
  <OMA>
    <OMS name="image" cd="fns1"/>
    <OMV name= "f"/>
  </OMA>
  <OMA>
```

```
    <OMS name="codomain" cd="fns1"/>
    <OMV name= "f"/>
  </OMA>
</OMBIND>
```

Indeed there could be (possibly even ought to be, since in ZF it is the *definition* of $\subset$) a FMP of `subset` that made this equivalent to expression 2.

3.5 C.2.2.15 MMLdefinition: `domainofapplication`

This contains the interesting statement

> Special cases of this qualifier can be abbreviated using one of interval `condition` or an (`lowlimit`,`uplimit`) pair.

The examples given are

```
<apply><int/>
  <domainofapplication><ci>C</ci></domainofapplication>
  <ci>f </ci>
</apply>
```

(which is a fairly straight-forward definite integral) and

```
<apply><int/>
      <domainofapplication>
        <set>
          <bvar><ci>t</ci></bvar>
          <condition>
            <apply><in/>
              <ci>t</ci>
              <ci type="set">C</ci>
            </apply>
          </condition>
        </set>
      </domainofapplication>
      <ci>f</ci>
</apply>
```

which seems to this author to be a redundant variant.

3.6 C.2.3.1 MMLdefinition: `quotient`

This has the following property (presumably meant to be the equivalent of OpenMath's FMP[19]).

[19] OpenMath's FMP for `quotient` in `integer1` is currently missing the proviso $b \neq 0$.

```
ForAll( [a,b], b != 0, a = b*quotient(a,b) + rem(a,b) )
```

```
<apply><forall/>
  <bvar><ci>a</ci></bvar>
  <bvar><ci>b</ci></bvar>
  <condition><apply><neq/><ci>b</ci><cn>0</cn></apply></condition>
  <apply><eq/>
    <ci>a</ci>
    <apply><plus/>
      <apply><times/>
          <ci>b</ci>
          <apply><quotient/><ci>a</ci><ci>b</ci></apply>
      </apply>
  <apply><rem/><ci>a</ci><ci>b</ci></apply>
    </apply>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though `forallrestricted` could be used.

```
<OMBIND>
  <OMS name="forall" cd="quant1"/>
  <OMBVAR> <OMV name="a"/> <OMV name="b"/> </OMBVAR>
  <OMA>
    <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="neq" cd="relation1"/>
      <OMV name="b"/>
      <OMS name="zero" cd="arith1"/>
    </OMA>
    <OMA>
      <OMS name="eq" cd="relation1"/>
      <OMV name="a"/>
      <OMA>
        <OMS name="plus" cd="arith1"/>
        <OMA>
          <OMS name="times" cd="arith1"/>
          <OMV name="b"/>
          <OMA>
            <OMS name="quotient" cd="integer1"/>
            <OMV name="a"/>
            <OMV name="b"/>
          </OMA>
        </OMA>
        <OMA>
          <OMS name="remainder" cd="integer1"/>
```

```
          <OMV name="a"/>
          <OMV name="b"/>
        </OMA>
      </OMA>
    </OMA>
  </OMA>
</OMBIND>
```

3.7 C.2.3.2 MMLdefinition: `factorial`

This has the following property (presumably meant to be the equivalent of OpenMath's FMP).

```
ForAll( n, n \gt 0, n! = n*(n-1)! )
```

```
<apply><forall/>
  <bvar><ci>n</ci></bvar>
  <condition><apply><gt/><ci>n</ci><cn>0</cn></apply></condition>
  <apply><eq/>
    <apply><factorial/><ci>n</ci></apply>
    <apply><times/>
      <ci>n</ci>
      <apply><factorial/>
        <apply><minus/><ci>n</ci><cn>1</cn></apply>
      </apply>
    </apply>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though `forallrestricted` could be used.

3.8 C.2.3.3 MMLdefinition: `divide`

This has the following property (presumably meant to be the equivalent of OpenMath's FMP).

```
ForAll( a, a!= 0, a/a = 1 )
```

```
<apply><forall/>
  <bvar><ci>a</ci></bvar>
  <condition><apply><neq/><ci>a</ci><cn>0</cn></apply></condition>
  <apply><eq/>
    <apply><divide/><ci>a</ci><ci>a</ci></apply>
    <cn>1</cn>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though `forallrestricted` could be used.

3.9 C.2.3.4 MMLdefinition: `max`

This contains the interesting statement

> The elements may be listed explicitly or they may be described by a `domainofapplication`, for example, the maximum over all x in the set A. The `domainofapplication` is often abbreviated by placing a `condition` directly on a bound variable.

The example given is the following (our layout).

```
<apply>
  <max/>
  <bvar><ci>y</ci></bvar>
  <condition>
    <apply>
      <in/>
      <ci>y</ci>
      <interval><cn>0</cn><cn>1</cn></interval>
    </apply>
  </condition>
  <apply><power/><ci> y</ci><cn>3</cn></apply>
</apply>
```

As OpenMath does not have a `max` operator acting on functions, the nearest translation would seem to be the following.

```
<OMA>
  <OMS name="max" cd="minmax1"/>
  <OMA>
    <OMS name="map" cd="set1"/>
    <OMBIND>
      <OMS name="lambda" cd="fns1"/>
      <OMBVAR> <OMV name="y"/> </OMBVAR>
      <OMA>
        <OMS name="power" cd="arith1"/>
        <OMV name="y"/>
        <OMI> 3 </OMI>
      </OMA>
    </OMBIND>
    <OMA>
      <OMS name="interval_cc" cd="interval1"/>
      <OMI> 0 </OMI>
      <OMI> 1 </OMI>
    </OMA>
```

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

We note that OpenMath seems to require us to be precise about the species of interval we want to use.

3.10 C.2.3.5 MMLdefinition: `min`

Nothing new need be said here.

3.11 C.2.3.7 MMLdefinition: `plus`

The property here is the following.

```
Commutativity
<apply><forall/>
  <bvar><ci>a</ci></bvar>
  <bvar><ci>b</ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci>a</ci><reals/></apply>
      <apply><in/><ci>b</ci><reals/></apply>
    </apply>
  </condition>
  <apply><eq/>
    <apply><plus/><ci>a</ci><ci>b</ci></apply>
    <apply><plus/><ci>b</ci><ci>a</ci></apply>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though `forallrestricted` could be used.

3.12 C.2.3.8 MMLdefinition: `power`

The property here is the following.

```
ForAll( a, a!=0, a^0=1 )
<apply><forall/>
  <bvar><ci>a</ci></bvar>
  <condition><apply><neq/><ci>a</ci><cn>0</cn></apply></condition>
  <apply><eq/>
    <apply><power/><ci>a</ci><cn>0</cn></apply>
    <cn>1</cn>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though `forallrestricted` could be used.

3.13 C.2.3.9 MMLdefinition: `rem`

This has the same property, and solution, as `quotient` (section 3.6).

3.14 C.2.3.10 MMLdefinition: `times`

The property here is the following.

```
ForAll( [a,b], condition(in({a,b}, Commutative)), a*b=b*a )
```

However, no formal translation is given, and it is not clear what one would be.

Later on we see the following property, to which the same remark applies as in section 3.11.

```
<apply><forall/>
  <bvar><ci>a</ci></bvar>
  <bvar><ci>b</ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci>a</ci><reals/></apply>
      <apply><in/><ci>b</ci><reals/></apply>
    </apply>
  </condition>
  <apply><eq/>
    <apply><times/><ci>a</ci><ci>b</ci></apply>
    <apply><times/><ci>b</ci><ci>a</ci></apply>
  </apply>
</apply>
```

3.15 C.2.3.18 MMLdefinition: `forall`

This contains the following example (our formatting).

```
<apply>
  <forall/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/><ci> x </ci><cn> 0 </cn></apply>
  </condition>
  <ci> x </ci>
</apply>
```

This seems to be $\forall x : x < 0x$. Since this last x is not a Boolean, the author respectfully submits that this example is ill-typed. In any case `forallrestricted` would solve the issue.

3.16 C.2.3.23 MMLdefinition: `real`

This contains the following property,

```
<apply><forall/>
  <bvar><ci>x</ci></bvar>
  <bvar><ci>y</ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci>x</ci><reals/></apply>
      <apply><in/><ci>y</ci><reals/></apply>
    </apply>
  </condition>
  <apply><eq/>
    <apply><real/>
      <apply><plus/>
        <ci> x </ci>
        <apply><times/><imaginaryi/><ci>y</ci></apply>
      </apply>
    </apply>
    <ci> x </ci>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though `forallrestricted` could be used.

3.17 C.2.5.1 MMLdefinition: `int`

This contains the following example.

```
<apply><int/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><in/><ci> x </ci><ci type="set"> D </ci></apply>
  </condition>
  <apply><ci type="function"> f </ci><ci> x </ci></apply>
</apply>
```

The discussion in section 3.3.2 is appropriate here.

3.18 C.2.5.6 MMLdefinition: `bvar`

This contains the following example (our formatting).

```
<apply><forall/><bvar><ci>x</ci></bvar>
  <condition><apply><in/><ci>x</ci><reals/></apply></condition>
  <apply>
```

```
    <eq/>
    <apply><minus/><ci>x</ci><ci>x</ci></apply>
    <cn>0</cn>
  </apply>
</apply>
```

Again, this seems to be a pretty good case for the "forall with implies" trick, though cforallrestricted ould be used.

3.19 C.2.5.8 MMLdefinition: divergence

This contains the following example.

```
<apply>
  <eq/>
  <apply><divergence/><ci type="vectorfield">a</ci></apply>
  <apply>
    <limit/>
    <bvar><ci> V </ci></bvar>
    <condition>
      <apply>
        <tendsto/>
        <ci> V </ci>
        <cn> 0 </cn>
      </apply>
    </condition>
    <apply>
      <divide/>
      <apply>
        <int encoding="text" definitionURL="SurfaceIntegrals.htm"/>
        <bvar><ci> S</ci></bvar>
        <ci> a </ci>
      </apply>
      <ci> V </ci>
    </apply>
  </apply>
</apply>
```

Here the relevant part is the limit, which could be expressed (as it is in the limit1 CD) as the following.

```
<OMA>
  <OMS cd="limit1" name="limit"/>
  <OMI> 0 </OMI>
  <OMS cd="limit1" name="both_sides"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
```

```
      <OMBVAR>
      <OMV name="V"/>
      </OMBVAR>
      ...
  </OMBIND>
</OMA>
```

3.20 C.2.6.1 MMLdefinition: `set`

This contains the following example.

```
<set>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply><lt/>
      <ci> x </ci>
      <cn> 5 </cn>
    </apply>
  </condition>
  <ci>x</ci>
</set>
```

It is not clear what this means, but a plausible stab would seem to be the following.

```
<OMA>
  <OMS cd="set1" name="suchthat"/>
  <OMS cd="setname1" name="N"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR>
      <OMV name="x"/>
    </OMBVAR>
    <OMA>
      <OMS cd="relation1" name="lt"/>
    <OMV name="x"/>
      <OMI> 5 </OMI>
    </OMA>
  </OMBIND>
</OMA>
```

3.21 C.2.6.2 MMLdefinition: `list`

This contains the following example.

```
<list order="numeric">
  <bvar><ci> x </ci></bvar>
```

```
  <condition>
    <apply><lt/>
      <ci> x </ci>
      <cn> 5 </cn>
    </apply>
  </condition>
</list>
```

There is no direct translation into OpenMath for reasons other than `condition`, but `sucthat` in `list1` seems an obvious tool to use.

3.22 C.2.6.7 MMLdefinition: `subset`

This contains the following example.

```
<apply>
  <subset/>
  <subset/>
  <bvar><ci type="set">S</ci></bvar>
  <condition>
    <apply><in/>
      <ci>S</ci>
      <ci type="list">T</ci>
    </apply>
  </condition>
  <ci>S</ci>
</apply>
```

Even assuming the second `<subset/>` to be a mistake, the present author can make no sense of this.

3.23 C.2.7.1 MMLdefinition: `sum`

This contains the following example (our formatting).

```
<apply><sum/>
  <bvar><ci> x </ci></bvar>
  <condition>
    <apply> <in/><ci> x </ci><ci type="set">B</ci></apply>
  </condition>
  <apply><ci type="function"> f </ci><ci> x </ci></apply>
</apply>
```

This translates straightforwardly.

```
<OMA>
  <OMS name="sum" cd="arith1"/>
  <OMV name="S"/>
```

```
  <OMV name="f"/>
</OMA>
```

We note that there is no need to name the dummy variable at all.

3.24 C.2.7.2 MMLdefinition: `product`

The example, and its OpenMath translation, are essentially identicalto the previous section.

3.25 C.2.7.3 MMLdefinition: `limit`

This contains the following example (our formatting).

```
<apply><limit/>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><tendsto/><ci>x</ci><cn>0</cn></apply>
  </condition>
  <apply><sin/><ci>x</ci></apply>
</apply>
```

The equivalent OpenMath would be the following.

```
<OMA>
  <OMS cd="limit1" name="limit"/>
  <OMI> 0 </OMI>
  <OMS cd="limit1" name="both_sides"/>
  <OMS cd="transc1" name="sin"/>
  </OMBIND>
</OMA>
```

We again note that there is no need to name the dummy variable at all.

3.26 C.2.10.2 MMLdefinition: `matrix`

This contains the following example (our formatting).

```
<matrix>
  <bvar><ci type="integer">i</ci></bvar>
  <bvar><ci type="integer">j</ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/>
        <ci>i</ci>
        <interval><ci>1</ci><ci>5</ci></interval>
      </apply>
      <apply><in/>
```

```
        <ci>j</ci>
        <interval><ci>5</ci><ci>9</ci></interval>
      </apply>
    </apply>
  </condition>
  <apply><power/>
    <ci>i</ci>
    <ci>j</ci>
  </apply>
</vector>
```

We can assume that this should end `</matrix>` instead of `</vector>`, but this has no equivalent in OpenMath.

3.27 C.2.11.3 MMLdefinition: `rational`

This contains the following property (our formatting).

```
    for all z where z is a rational, there exists
      integers p and q with p/q = z
<apply><forall/>
  <bvar><ci>z</ci></bvar>
  <condition>
    <apply><in/><ci>z</ci><rationals/></apply>
    </condition>
  <apply><exists/>
    <bvar><ci>p</ci></bvar>
    <bvar><ci>q</ci></bvar>
    <apply><and/>
      <apply><in/><ci>p</ci><integers/></apply>
      <apply><in/><ci>q</ci><integers/></apply>
      <apply><eq/>
        <apply><divide/><ci>p</ci><ci>q</ci></apply>
        <ci>z</ci>
      </apply>
    </apply>
  </apply>
```

`forallrestricted` seems the obvious solution, though the implies trick could also be used.

3.28 C.2.11.6 MMLdefinition: `primes`

This contains the following property (our formatting).

```
ForAll( [d,p], p is prime, Implies( d | p , d=1 or d=p ) )
<apply><forall/>
  <bvar><ci>d</ci></bvar>
```

```
  <bvar><ci>p</ci></bvar>
  <condition>
    <apply><and/>
    <apply><in/><ci>p</ci><primes/></apply>
    <apply><in/><ci>d</ci><naturalnumbers/></apply>
    </apply>
  </condition>
  <apply><implies/>
    <apply><factorof/><ci>d</ci><ci>p</ci></apply>
    <apply><or/>
      <apply><eq/><ci>d</ci><cn>1</cn></apply>
      <apply><eq/><ci>d</ci><ci>p</ci></apply>
    </apply>
  </apply>
</apply>
```

This could be encoded with the forall/implies trick, except that the result would have two implication signs — perfectly correct, but possibly harder to read. We would need to nest `forallrestricted`, as in the following.

```
<OMBIND>
  <OMA>
    <OMS name="forallrestricted" cd="quant2"/>
    <OMS name="N" cd="setname1"/>
  </OMA>
  <OMBVAR> <OMV name="d"/> </OMBVAR>
  <OMBIND>
    <OMA>
      <OMS name="forallrestricted" cd="quant2"/>
      <OMS name="P" cd="setname1"/>
    </OMA>
    <OMBVAR> <OMV name="p"/> </OMBVAR>
    ...
  </OMBIND>
</OMBIND>
```

3.29 C.2.11.15 MMLdefinition: `infinity`

This contains the following property and example (our formatting).

3.29.1 C.2.11.15(1) MMLdefinition: `infinity`

```
    for all reals x, x \lt infinity
<apply><forall/>
  <bvar><ci>n</ci></bvar>
  <condition><apply><in/><ci>n</ci><reals/></apply></condition>
  <apply><lt/><ci>n</ci><infinity/></apply>
```

```
</apply>
```

`forallrestricted` seems the obvious solution, though the implies trick could also be used.

3.29.2 C.2.11.15(2) MMLdefinition: `infinity`

```
<apply><eq/>
  <apply><limit/>
    <bvar><ci>x</ci></bvar>
    <condition>
      <apply><tendsto/><ci>x</ci><infinity/></apply>
      </condition>
    <apply><divide/><cn>1</cn><ci>x</ci></apply>
  </apply>
  <cn>0</cn>
</apply>
```

From OpenMath's point of view, this is a straightforward limit.

```
<OMA>
  <OMA>
    <OMS cd="limit1" name="limit"/>
    <OMS name="infinity" cd="nums1"/>
    <OMS cd="limit1" name="below"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR> <OMV name="x"/> </OMBVAR>
      <OMA>
        <OMS name="divide" cd="arith1"/>
        <OMI> 1 </OMI>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
  <OMS cd="alg1" name="zero"/>
</OMA>
```

References

[1] Davenport,J.H., OpenMath in a (Semantic) Web. Presentation at third Joining Educational Mathematics Workshop `http://www.jem-thematic.net/files_private/Barcelona.pdf`, February 1, 2008.

[2] Davenport,J.H., OpenMath and MathML: Differentiating between analysis and algebra. `http://staff.bath.ac.uk/masjhd/differentiate.html`, October 4, 2008.

[3] Kohlhase,M., OpenMath3 without conditions: A Proposal for a MathML3/OM3 Calculus Content Dictionary. `http://svn.openmath.org/OpenMath3/doc/blue/noconds/note.tex`, September 6, 2008.

Table 1: `condition` in Chapter 4

MathML 2 Chapter 4	heading	This document	resolution
4.2.1.8	qualifiers	2.1.1	`suchthat`
		2.1.2	`map`
4.2.3.2	operators with qualifiers	2.3.1	Needs work
		2.3.2	`suchthat`
		2.3.3	no solution
		2.3.4	`forallrestricted`[1]
4.2.5	Conditions	2.4.1	none needed
		2.4.2	`forall/existsrestricted`
		2.4.3	`existsrestricted`[1]
4.4.2.7.2	Conditions	2.5.3	`suchthat`
4.4.3.4	Maximum	2.6.1	`map`?[2]
		2.6.2	`suchthat`
4.4.3.17	`forall`	2.7.1	`forallrestricted`[3]
		2.7.2	`forallrestricted`[3]
4.4.5.1	`int`	2.8	Needs work
4.4.5.6.1	`bvar`	2.9	`suchthat`
4.4.5.6.2	`bvar`	2.10	Needs work
4.4.6.1.2	`set`	2.11	`suchthat`
4.4.6.2.2	`list`	2.12	`integer_interval`
4.4.6.7	Subset	2.13	loose
4.4.7.1	Sum	2.14	None needed
4.4.7.2	Product	2.15	None needed
4.4.7.3	Limit	2.16	Use `limit1` CD

Notes

1. or nothing special.
2. the query is caused by the fact that this fragment is close to meaningless.
3. but `forallrestricted` doesn't do a perfect job.

Table 2: `condition` in Appendix C

MathML 2 Appendix C	heading	This document	resolution
C.2.2.4	`interval`	section 3.1	`suchthat`; ??
C.2.2.5	`inverse`	section 3.2	`forallrestricted`
C.2.2.7	`condition`	section 3.3.1	`suchthat`
C.2.2.7	`condition`	section 3.3.2	Needs work
C.2.2.14	`image`	section 3.4	`forallrestricted`[1]
C.2.2.15	`domainofapplication`	section 3.5	none needed
C.2.3.1	`quotient`	section 3.6	`forallrestricted`[1]
C.2.3.2	`factorial`	section 3.7	`forallrestricted`[1]
C.2.3.3	`divide`	section 3.8	`forallrestricted`[1]
C.2.3.4	`max`	section 3.9	`map`
C.2.3.5	`min`	section 3.10	`map`
C.2.3.7	`plus`	section 3.11	`forallrestricted`[1]
C.2.3.8	`power`	section 3.12	`forallrestricted`[1]
C.2.3.9	`quotient`	section 3.13	As section 3.6
C.2.3.10	`times`	section 3.14	No formal MathML; As section 3.11
C.2.3.18	`times`	section 3.15	Apparently meaningless
C.2.3.23	`real`	section 3.16	`forallrestricted`[1]
C.2.5.1	`int`	section 3.17	As section 3.3.2
C.2.5.6	`bvar`	section 3.18	`forallrestricted`[1]
C.2.5.8	`divergence`	section 3.19	Use `limit1` CD
C.2.6.1	`set`	section 3.20	`suchthat`
C.2.6.2	`list`	section 3.21	`suchthat/list1`
C.2.6.7	`subset`	section 3.22	Apparently meaningless
C.2.7.1	`sum`	section 3.23	`sum`
C.2.7.2	`product`	section 3.24	`product`
C.2.7.3	`limit`	section 3.25	`limit`
C.2.10.2	`matrix`	section 3.26	No equivalent
C.2.11.3	`rational`	section 3.27	`forallrestricted`
C.2.11.6	`primes`	section 3.28	`forallrestricted`[1]
C.2.11.15	`infinity`	section 3.29.1	`forallrestricted`[1]
C.2.11.15	`infinity`	section 3.29.2	Use `limit1` CD

[1] or nothing special