

# Conditions in MathML

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

October 11, 2008

## Abstract

JHD's attempt (currently very incomplete) to look at the uses of `condition` in the MathML2 specification.

## 1 Introduction

This note follows from the OpenMath tele-conference at 15:00 GMT<sup>1</sup> 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.

## 2 Appendix C

### 2.1 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>
```

---

<sup>1</sup>16:00 BST, 17:00 CET.

Presumably this represents  $(0, \infty)$ . Equally, this is presumably intended to close over `<interval>`, i.e.  $x$  is bound in this expression, and freely  $\alpha$ -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.

## 2.2 C.2.2.5 MMLdefinition: inverse

This contains the following example.

```
<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="quantifier2"/>
    <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  $\alpha$ -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 Open-Math2 standard restricts alpha-conversion to the second and third children of the OMBIND, which is consistent with this view.

See section 2.3.2 below for an example of where this rule bites.

## 2.3 C.2.2.7 MMLdefinition: condition

This section contains two examples.

### 2.3.1 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.

```
<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.

### 2.3.2 C.2.2.7(2) MMLdefinition: condition

```
<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:

$$\int_{x \in C} f(x) dx, \tag{1}$$

and observe that this is probably equivalent to  $\int_C f(x) dx$ .

In terms of `calculus1` (calculus with functions [1]) 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 [2] on `calculus3` (calculus with expressions [1]), 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.

```
<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 (1), which in MathML syntax would be the following.

```
<bind>
  <apply>
    <csymbol cd="calculus3">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>
  </OMA>
  <OMBVAR>
    <OMV name="x"/>
  </OMBVAR>
  <OMA>
    <OMV name="f"/>
  </OMA>
```

```
<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 `definitcond` as postulated can make its way into OpenMath.

## References

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