Quantifiers and Big Operators in OpenMath

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. The effort to align MathML 3 and OpenMath has led to a realisation that (pragmatic) MathML's `condition` and `domainofapplication` elements, when used with quantifiers, do not have a *neat* expression in OpenMath.
This paper analyzes the situation focusing on quantifiers and proposes a solution, via six new symbols. Two of them fit completely within the existing OpenMath structure, and we place them in the associated `quant3` CD. The others require a generalization of `OMBIND`.
We also propose, logically separately but in the same area, a `quant2` CD with `existsuniquely`, commonly written ∃!, and the 'fusion' symbol `existsuniquelyin`.
In a second step we generalize the solution to the phenomenon of big operators that MathML 2 implicitly provides but which do not have a direct counterpart in the OpenMath CDs.

1 Introduction

The effort to align MathML 3 [Con08] and OpenMath [Con04] has led to a realisation that (pragmatic) MathML 3's `condition` and `domainofapplication` elements, when used with quantifiers, does not have a *neat* expression in OpenMath. We have described the MathML 3/OpenMath 3 and the general representational issues in [DK09], which we will assume as background reference. But a central part of the alignment effort remains unsolved there: the provisioning of the OpenMath content dictionaries that supply the necessary symbols. We will describe what needs to be done on this front in this paper, which should be seen as a companion paper to [DK09].

We start out by analyzing the situation focusing on quantifiers and then generalizing the solution to the phenomenon of big operators that MathML 2 implicitly provides but which do not have a direct counterpart in the OpenMath CDs.

2 Existential and Universal Quantifiers

"$\forall n \in \mathbb{N} \ldots$" is a very natural piece of mathematical notation, even though it tends not to be formally defined. MathML (Content) can represent this via the

`condition` element, but OpenMath[3] has hitherto had no *direct* way of doing so, relying on the following equivalences:

$$\forall v \in S.p(v) \Leftrightarrow \forall v.(v \in S) \Rightarrow p(v) \quad (1)$$

$$\exists v \in S.p(v) \Leftrightarrow \exists v.(v \in S) \wedge p(v) \quad (2)$$

However, in practice[4], it would be better to have a convenient shorthand for these, hence this proposal.

The challenge is the syntax of OpenMath. One represents $\forall n.p(n)$ as

```
1 <OMBIND>
    <OMS name="forall" cd="quant1"/>
    <OMBVAR><OMV name="n"/></OMBVAR>
    <OMA><OMV name="p"/><OMV name="n"/></OMA>
  </OMBIND>
```

Where is the "$\in \mathbb{N}$" going to fit in this syntax? In [DK09] we have argued that there are actually two representation styles that need to be supported: Figure 1 gives both styles the *first-order* style on the left makes use of expressions with bound variables whereas the *higher-order* style directly uses sets. The latter is

<OMBIND> <OMS name="forallcond" cd="quant3"/> <OMBVAR> <OMV name="n"/> </OMBVAR> <OMA> <OMS name="in" cd="set1"/> <OMV name="n"/> <OMS name="N" cd="setname1"/> </OMA> $\boxed{p(n)}$ </OMBIND>	<OMBIND> <OMA> <OMS name="forallin" cd="quant3"/> <OMS name="N" cd="setname1"/> </OMA> <OMBVAR> <OMV name="n"/> </OMBVAR> $\boxed{p(n)}$ </OMBIND>
here and in the following we use boxed mathematical notation to represent the obvious OpenMath counterparts	

Fig. 1. Two representations of quantifiers with domains

nearer to our example, and we propose to use the fact that the first child of an `OMBIND` need not be an atom and propose a symbol `forallin` which acts as a binding constructor, i.e. an operator that takes a set as an argument and returns a binding operator, which can then be used in the `OMBIND`. The first-order style of representation could be glossed in Mathematics as $\forall_c n : n \in \mathbb{N}.p(n)$; it makes the bound variable that was implicit in the higher-order construction explicit in a condition expression. So we would contend that the higher-order style is closer to our original example. But the first-order style is more in other situations, e.g. $\forall x : (1/x \text{ is irrational}).p(x)$, which can be represented as $\boxed{\forall x : (\forall n, d.n/d \neq x).p(x)}$.

[3] If restricted only to the content dictionaries the authors know about.

[4] A referee objected to [DK09] on the grounds that it had stated that such shorthand was not necessary, writing 'one might [as well] write "not p" as"p implies false"'.

Listing 1.1. A natural case for a quantifier with a condition

```
<OMBIND>
  <OMS name="forallcond" cd="quant3"/>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMBIND>
    <OMS name="forall" cd="quant1"/>
    <OMBVAR><OMV name="n"/><OMV name="d"/></OMBVAR>
    <OMA><OMS cd="relation1" name="ne"/>
      <OMA><OMS cd="arith1" name="divide">
        <OMV name="n"/>
        <OMV name="d"/>
      </OMA>
      <OMV name="x"/>
    </OMA>
  </OMBIND>
  p(x)
</OMBIND>
```

Note that for the first-order representations on the left of Figure 1 and in Listing 1.1 we need the extension of binding objects to allow multiple scopes proposed in [DK09]. We consider the examples in this paper and representation possibilities they enable to be a good reason for this extension.

The symbols `forallin` and `forallcond` are defined in the proposed `quant3` CD, with Formal Mathematical Properties corresponding to equations (1) and (2), at least in the single-variable case. They are related by the following FMPs:

$$\begin{aligned} \forall P, Q.\ [\forall\!\!\!_{c} x.Q(x)P(x)] &\Leftrightarrow [\forall\!\!\!_{i}(\lambda z.P(z))]x.Q(x) \\ \forall Q, S.\ [\forall\!\!\!_{i}(S)]x.Q(x) &\Leftrightarrow \forall x.(x \in S) \Rightarrow Q(x) \end{aligned}$$

where we use $\forall\!\!\!_{c}$ for `forallcond` and $\forall\!\!\!_{i}$ for `forallin`. We should note what `forallin` does, and does not, encode.

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

does not $\forall n \in \mathbb{N}, x \in \mathbb{R}$ (this needs two nested bindings of `forallin`[5]), $\forall n > 2$ (though we *can* encode $\forall n \in (2,\infty)$), $\forall m < n \in \mathbb{N}$ (though we *can* encode $\forall n \in (-\infty, n)$ and use `integer_interval`).

[Con03, section 4.2.5.1] gives an example for the formula

$$\forall x \in \mathbb{N}.\exists p, q \in \mathbb{P}.p + q = 2x$$

where $\mathbb{P}$ stands for the set of prime numbers. While this would succumb to forall/implies and exists/and encodings, it is a better tribute to the power of our extended quantifiers to use the following encoding:

```
<OMBIND>
  <OMA>
    <OMS name="forallin" cd="quant3"/>
    <OMS name="N" cd="setname1"/>
  </OMA>
```

[5] At first sight it seems that this could be represented as $\forall\!\!\!_{i}\mathbb{N} \times \mathbb{R}$. but we contend that these are different. We can encode $\forall z \in (\mathbb{N} \times \mathbb{R})$, and *later* destructure z, but OpenMath doesn't have a destructuring bind.

```
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMBIND>
    <OMA>
      <OMS name="existsin" cd="quant3"/>
      <OMS name="P" cd="setname1"/>
   </OMA>
    <OMBVAR><OMV name="p"/><OMV name="q"/></OMBVAR>
    p + q = 2x
  </OMBIND>
</OMBIND>
```

Note that in some cases, we naturally combine these two: We often see something like the following $\forall x, y \in \mathbb{R} : x - y \neq 0 . \frac{1}{x-y} \in \mathbb{R}$. This would be suitably encoded as

```
<OMBIND>
  <OMA>
    <OMS name="forallincond" cd="quant3"/>
    ℝ
  </OMA>
  <OMBVAR><OMV name="x"/><OMV name="y"/></OMBVAR>
  1/(x−y) ∈ ℝ
  x − y ≠ 0
</OMBIND>
```

using the a symbol `forallincond` that we propose to add to `quant3` content dictionary together with the `FMP`

$$\forall_{ic}(S)x : C(x).D \Leftrightarrow \forall_i(S).C(x) \Rightarrow D$$

The existential variants `existin`, `existscond`, and `existsincond` of all three have analogous FMPs in the `quant3` content dictionary.

3 Unique Existence

Although the notation $\exists!$ is relatively new[6], it is convenient. It would be easy enough to introduce into OpenMath, as a symbol, say `existsuniquely` in the `quant2` CD, with one property, which could be held to define it.

$$\exists!x.p(x) \Leftrightarrow (\exists x.p(x)) \wedge ((p(x) \wedge p(y)) \Rightarrow x = y)\,. \tag{3}$$

One might naturally ask: "what about $\exists!x \in \mathbb{N} : \ldots$, and similar constructs". There is obviously no difficulty (other than the length of the name!) in adding `existsuniquelyin` to the `quant3` CD. The real challenge is what semantics does one want. There are two options.

"It's unique *and* **it's in $\mathbb{N}$"**

$$\exists!x \in \mathbb{N}.p(x) \Leftrightarrow (\exists x.(x \in \mathbb{N} \wedge p(x)) \wedge ((p(x) \wedge p(y)) \Rightarrow x = y)\,. \tag{4}$$

This is what one would get by applying equation (2) to the $\exists$ on the right-hand side of equation (3).

[6] It was not in use when the first author was a student, and the earliest we can trace it to is [Bar84, p. xiii].

"It's unique *within* $\mathbb{N}$**"**

$$\exists! x \in \mathbb{N}.p(x) \Leftrightarrow (\exists x.(x \in \mathbb{N} \wedge p(x))) \wedge (((p(x) \wedge x \in \mathbb{N}) \wedge (p(y) \wedge y \in \mathbb{N})) \Rightarrow x = y)\,. \tag{5}$$

This is what one would get by applying equation (3) to equation (2).

While both have their part to play, it appears that (5) is the one more commonly met, and we propose to add this meaning to `quant3`.

4 Other Similar cases: Big Operators

Note that in the analysis above, the fact that we are dealing with quantifiers is secondary, the interesting part is the fact that, the "operators take qualifiers" `condition` and `domainofapplication` in MathML 2. The full list of these operators can be grouped into three parts: the special operators `int`, `sum`, `product`, `root`, `diff`, `partialdiff`, `limit`, `log`, `moment`, `forall`, `exists`, the binary/$n$ary operators `plus`, `times`, `max`, `min`, `gcd`, `lcm`, `mean`, `sdev`, `variance`, `median`, `mode`, `and`, `or`, `xor`, `union`, `intersect`, `cartesianproduct`, `compose`, as well as the relational operators `eq`, `leq`, `lt`, `geq`, `gt`. The use of qualifiers with relational operators is being deprecated in MathML 3, so we will not concern ourselves with these.

The reason binary operators can take qualifiers is that they can be "lifted" to their respective "big operator form", for instance $\bigcup$ for $\cup$. Note that if we look at the special operators in the first group, then we can see that `sum`, `product`, `forall`, and `exists` are the conventionalized "big operators" for `plus`, `times`, `and`, and `or`. We have covered the quantifiers above, so let us look whether `sum` and `product` show the same behavior. If they do, the quantifiers may give a good model for the other "big operators". Here we can directly see that all cases occur, witnessed e.g. by the Lagrange base polynomial $L(x) := \Pi_{i=0,i\neq j}^{k} \frac{x - x_i}{x_j - x_i}$, which would need a symbol `productincond` in `arith2`.

5 Other Operators that take Qualifiers

There are other operators that take qualifiers in MathML; these are amenable to the same treatment: Consider the naive set $\{x^2 | x < 1\}$, which could be represented in MathML 2 as

```
<set>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><lt/><ci>x</ci><cn>1</cn></apply>
  </condition>
  <apply><power/><ci>x</ci><cn>2</cn></apply>
</set>
```

We propose a `suchthatcond` binding constructor that would allow us to write

```
<OMBIND>
  <OMS cd="set4" name="suchthatcond"/>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMA>
    <OMS cd="relation1" name="lt"/>
    <OMV name="x"/>
    <OMI>1</OMI>
  </OMA>
  <OMA>
    <OMS cd="arith1" name="power"/>
    <OMV name="x"/>
    <OMI>2</OMI>
  </OMA>
</OMBIND>
```

6 Conclusion

We have studied the representation of extended quantifiers like $\forall x \in S.$ or $\forall x : p(x).$ commonly found in informal mathematical texts. While it is possible to encode these in principle with the quantifiers and connectives provided by the `quant1` and `logic1` content dictionaries from the MathML CD group. The encoding loses the surface structure of the original mathematical expressions and does not support the same intuitive modes of reasoning. Therefore we propose to augment the OpenMath society's set of standard CDs with a new content dictionary `quant3` with six new symbols and `FMP`s that relate them to the classical quantifiers.

By the same concerns for structural adequacy we propose to introduce a symbol for unique existence to the existing `quant2` content dictionary and various operators for "lifted operators" that MathML 2 implicitly supports by allowing them to "take qualifiers".

We feel that the proposed symbols will make the use of OpenMath more intuitive and thus make OpenMath more attractive as a whole for the working mathematician. At the same time, people who prefer the classical quantifiers can refrain from using the new symbols.

It should be noted $\exists!$, $\exists_i$ and $\forall_i$ do *not* require any changes to OpenMath: the rest require an extension of the concept of binding, or the acceptance of mathematically meaningless 'gluing' operators to allow for the fact that bding should take place over both the body and the predicate.

Hence our proposals, in increasing order of scope, are as follows.

1. Adopt `existsuniquelyin`, i.e. $\exists!$.
2. Adopt `existsin`, i.e. $\exists_i$, and`forallin`, i.e. $\forall_i$.
3. Accept that `OMBIND` should be able to bind over more than one child.
4. Adopt `forallcond` etc.
5. Adopt `suchthatcond`, and other related symbols.

Should 3 prove a bridge too far, the subsequent proposals *could* be adopted with a mathematically meaningless gluing operator, as in [DK09].

References

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

[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.

[Con04] The OpenMath Consortium. OpenMath Standard 2.0. `http://www.openmath.org/standard/om20-2004-06-30/omstd20.pdf`, 2004.

[Con08] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 3.0: W3C Working Draft 17 November 2008. `http://www.w3.org/TR/2008/WD-MathML3-20081117`, 2008.

[DK09] J.H. Davenport and M. Kohlhase. Unifying Math Ontologies: A tale of two standards (extended abstract). In L. Dixon *et al.*, editor, *Proceedings Calculemus/MKM 2009*, pages 263–278, 2009.