On the Conversion between Content MathML and OpenMath

Clare M. So, Stephen M. Watt

Content MathML and OpenMath are two XML formats for semantic markup of mathematical expressions. Although efforts have been made to align their definitions, enough differences remain to make translation between them nontrivial.

In this chapter, we present a technical discussion of the differences between Content MathML and OpenMath, and two strategies for bidirectional translation between them. The first approach is to map between predefined Content MathML elements and standard OpenMath definitions, making any necessary adaptations for concepts that do not exactly correspond. For the second method, we describe a mapping using the extension mechanisms of each language, wherein all references to standard concepts are replaced by equivalent external references.

We have implemented these translation schemes (both approaches and both directions) as XSLT stylesheets. These implementations have served both to test our ideas and as components in an architecture for mathematical web services.

1 Introduction

Mathematical software packages, including computer algebra systems and theorem provers, define their own system-dependent syntax for commands and expressions. This is appropriate under the assumption that a person uses only a small number of such systems. The drawback of package-specific notation for mathematical content is that it makes data exchange between programs difficult and it limits a person's ability to take advantage of the strengths of unfamiliar systems.

OpenMath [2, 9, 10] and Content MathML [4] were both devised to provide system-independent, non-proprietary formats for the semantics of mathematical expressions. They are complementary standards with different emphases. For example, OpenMath currently does not define how its expressions should be displayed, and can use MathML for that purpose. Conversely, Content MathML may be extended using OpenMath annotations to express semantic concepts that it does not natively provide.

The relationship between OpenMath and MathML has been discussed elsewhere [6, 7, 11].

There are two implications to the fact that the standards are complementary. First, since each standard may rely to some extent on the other, it is clear that the languages are needed simultaneously for the general management of mathematical knowledge (on the web or otherwise). Hence, translations between the languages are necessary. However, the co-dependence between the languages also implies that possibly they are different on a deeper level, that is, sufficiently mismatched so as to complicate such translations. Indeed, as we will describe, there are differences which prevent the simple translation of certain expressions from conserving semantics.

Our primary aim in this chapter is to give a detailed analysis of translation between the Content MathML and OpenMath formats. In doing so, we uncover certain discrepancies that may preclude the full interoperability of the languages, and which therefore should be taken into consideration in future versions of these standards.

We describe two strategies for translation between Content MathML and OpenMath, and we demonstrate how these mismatches may be addressed in both cases. The first strategy is to convert between pre-defined Content MathML elements and OpenMath's MathML *Content Dictionaries*, preserving the use of Content MathML's built-in semantics if possible. This translation also handles the primitives of both formats. The second strategy maps all semantics of the source into expressions using the extension mechanism of the target language. This open-ended strategy uses only the low-level expression forming primitives of OpenMath and Content MathML, and uses external definitions for all mathematical content. Both strategies match the basic elements of the standards.

To summarize, the principle contributions of this work are:

- *A strategy for high-level conversion between OpenMath and Content MathML.* We identify mismatches between Content MathML and OpenMath, including shortcomings of the OpenMath "MathML" Content Dictionary, intended to support Content MathML.
- *A strategy for uniform conversion between OpenMath and MathML.* The target language, either OpenMath or Content MathML, is used as a "carrier" syntax, and translations uniformly use the extension mechanisms.

The remainder of this chapter is organized as follows. Section 2 introduces the Content MathML and OpenMath standards. Section 3 presents the mapping between the semantics of Content MathML and OpenMath's MathML CD group. It also presents the discrepancies between Content

MathML and OpenMath. Section 4 describes how we can always use external definitions in Content MathML when matching semantics are not available. This translation strategy helps us to overcome the cases in which OpenMath's semantics do not have Content MathML equivalent. Section 5 describes our implementation of a translator based on these ideas and Section 6 concludes the chapter.

2 OpenMath and Content MathML

The OpenMath and Content MathML standards both provide facilities to manage mathematical knowledge. Both formats serve to encode the semantics of mathematical expressions. Neither format is used to specify mathematical notation. Rather, it is expected that TEX or *Presentation MathML* can be used to represent expression notation, which is beyond the scope of this chapter. Before contrasting the two languages, we highlight their key characteristics individually.

2.1 OpenMath. OpenMath a system-independent XML [5] format to encode the semantics of mathematical expressions. It can be used as input and output for scientific computation as well as the format for mathematical expressions in documents. This standard includes a set of predefined elements describing some elementary concepts such as variables, floating point numbers and integers. No mathematical function is predefined. All semantics of the mathematical functions are defined in collections called *Content Dictionaries* (CDs). These CDs may be defined by a single application, by an agreement between two parties or by community consensus.

Example 1.1 $\int x^2\,dx$ in OpenMath

```
<OMOBJ xmlns="http://www.openmath.org/">
  <OMA>
    <OMS cd="calculus1" name="int"/>
    <OMBVAR> <OMV name="x"/> </OMBVAR>
    <OMA>
      <OMS cd="arith1" name="power"/>
      <OMV name="x"/> <OMI> 2 </OMI>
    </OMA>
  </OMA>
</OMOBJ>
```

§ Extending OpenMath. OpenMath is extensible by design. The semantics of mathematical functions are defined using Content Dictionaries, which may, in principle, be contributed by anyone. This removes the need to refer to semantics expressed in languages other than OpenMath. Existing, frequently-used mathematical functions and constants are pre-defined in official "standard" OpenMath CDs.

CD name	Description
alg1	Some basic algebraic concepts
arith1	Common arithmetic functions
bigfloat1	Representation of floating point numbers
calculus1	Calculus operations
complex1	Operations and constructors of complex numbers
fns1	Constructors and functions for functions
integer1	Basic integer functions
interval1	Discrete and continuous 1-dimensional intervals
linalg1	Operations on matrices
linalg2	Matrices and vectors in a row oriented fashion
limit1	Basic notion of the limits of unary functions
list1	List constructors
logic1	Basic logic functions and constants
mathmltypes	Types and constructs handled by MathML
minmax1	Minimum and maximum of a set
multiset1	Basic multiset theory
nums1	Common numerical constants and constructors
piece1	Operators for piece-wise defined expressions
quant1	Basic universal and existential quantifiers
relation1	Common arithmetic operations
setname1	Common sets in mathematics
rounding1	Basic rounding concepts
set1	Functions and constructors for basic set theory
s_data1	Basic statistical functions used on sample data
s_dist1	Basic statistical functions used on random variables
transc1	Transcendental functions
veccalc1	Functions for vector calculus
altenc	Alternative encodings

Figure 2.1. OpenMath's MathML CD group.

The standard CDs and CD groups of OpenMath are organized so that related functions and constants are usually placed in the same CD. Related CDs then belong to a CD group. The OpenMath Society has already approved a certain number of CDs and CD groups. In the next section we examine one of the OpenMath Society's official CDs, the MathML CD group, shown in Figure 2.1.

2.2 Content MathML.

MathML [4] is a system-independent format to encode mathematical expressions for web documents. Different types of MathML markup are available to accommodate different applications of mathematics. *Presentation markup* provides a format for specifying the notation of a mathematical expression. *Content markup* provides a format for specifying the semantics and the structure of mathematical expressions without implying any actual notation being used. This type of markup allows applications to specify the notation, and process the semantics of the expression separately. *Mixed markup* associates the two types of markup by bundling together, at some level of resolution, the notation and the semantics of a mathematical expression. This type of markup allows a variety of applications to process the expression, using either the semantics or the notation.

Since we aim to convert between OpenMath and MathML, we are concerned only with the Content markup aspect of MathML. Currently, Content MathML consists of a base set of elements covering subjects in elementary mathematics. The subject areas covered by Content MathML to some extent include arithmetic, algebra, logic, relations, calculus, set theory, sequences and series, elementary classical functions, statistics and linear algebra.

Example 1.2 $\int x^2\,dx$ in Content MathML

```
<math xmlns="http://www.w3.org/Math/">
  <apply>
    <int/>
    <bvar> <ci> x </ci> </bvar>
    <apply> <power/> <ci>x</ci> <cn>2</cn> </apply>
  </apply>
</math>
```

§ Extending Content MathML. Content MathML's base set of elements cannot natively handle more than a fraction of the vast set of mathematical concepts. To overcome this problem, Content MathML provides an extension mechanism allowing one to redefine Content MathML elements or to introduce new mathematical semantics. The following examples illustrate the use of these extension mechanisms in Content MathML.

Example 1.3 Content MathML predefines `<minus>` as a unary or a binary arithmetic operator. If we would like to use this element as a n-ary operator, we have to override its semantics. This is accomplished by referring to an external URL. The `definitionURL` attribute gives a location defining the new semantics. The following Content MathML markup illustrates how one might encode the expression $1 - 2 - 3 - 4 - 5$:

```
<apply>
  <minus definitionURL=
         "http://www.orcca.on.ca/example/MathML/n-ary#new_minus"/>
  <cn>1</cn> <cn>2</cn> <cn>3</cn> <cn>4</cn> <cn>5</cn>
</apply>
```

A second example shows how to introduce completely new mathematical functions into Content MathML. The <csymbol> element is used for this purpose.

Example 1.4 Bessel functions of the first kind are not defined in Content MathML. To introduce these functions, we need to specify a URL giving the semantics as an attribute within a <csymbol> element. The following Content MathML markup illustrates how to define and use the function:

```
<apply>
   <csymbol definitionURL=
     "http://www.orcca.on.ca/example/OpenMath/SpecialFns/bessel#BesselJ"
     >BesselJ</csymbol>
   <cn> 0 </cn>
   <cn> 0 </cn>
</apply>
```

3 Conversion Using Internal Semantics

On first examination, Content MathML and OpenMath appear quite similar. One-to-one correspondences can be drawn between most of the basic elements and between some mathematical elements as well. This is shown in the example in Figure 3.2. For example, <apply> maps to <OMA>, <ci> maps to <OMV>, and the <OMS>s map to their corresponding MathML mathematical functions.

```
<math>
  <apply>
    <equivalent/>
    <apply>
      <not/>
      <apply>
        <and/>
        <ci> A </ci>
        <ci> B </ci>
      </apply>
    </apply>
    <apply>
      <and/>
      <apply>
        <not/>
        <ci> A </ci>
      </apply>
      <apply>
        <not/>
        <ci> B </ci>
      </apply>
    </apply>
  </apply>
<math>
```

```
<OMOBJ>
  <OMA>
    <OMS cd="logic1" name="equivalent"/>
    <OMA>
      <OMS cd="logic1" name="not"/>
      <OMA>
        <OMS cd="logic1" name="and"/>
        <OMV name="A"/>
        <OMV name="B"/>
      </OMA>
    </OMA>
    <OMA>
      <OMS cd="logic1" name="and"/>
      <OMA>
        <OMS cd="logic1" name="not"/>
        <OMV name="A"/>
      </OMA>
      <OMA>
        <OMS cd="logic1" name="not"/>
        <OMV name="B"/>
      </OMA>
    <OMA>
  </OMA>
</OMOBJ>
```

Figure 3.2. Content MathML (left) and OpenMath (right) markup of $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$

The OpenMath MathML CD group is designed to provide MathML compatibility. It is intended to enable simple translation, mapping between functions described by this CD group and Content MathML elements. Such a translation is reported in [12].

Translation between these two formats is not always so simple, however. Although OpenMath's MathML CD group is designed to be compatible with Content MathML, some mismatches between OpenMath and Content MathML exist. Specifically, some elements present in one language are not present in the other, and some elements present in both languages have slightly different meanings or usage. A robust translator must deal with differences in the meanings of the pre-defined mathematical functions as well as expressions that make use of the extension mechanisms in both languages.

3.1 Dealing with the Differences.

There are three ways to handle differences between Content MathML and OpenMath within a translation: If usage is different between two matching elements, a special transformation can be performed. In cases where there is more than one possible match, heuristics can be derived for eliminating inappropriate choices. Finally, failing these two options, we can extend Content MathML by making external reference to OpenMath's semantics.

§ Special Transformation. Although there is some overlap between the development communities of OpenMath and Content MathML, differences in usage can be found in matching language elements. In such cases, we can construct the mathematical object based on the information within the existing markup. This solution can be used in both directions of the translation. Example 1.5 illustrates this approach.

Example 1.5 Intervals are expressed differently in Content MathML and OpenMath. This example shows the interval $(0, \pi)$ in Content MathML (left) and OpenMath (right). Knowing the difference between the two formats, we simply perform a special transformation to reconstruct the appropriate elements for either target.

```
<interval closure="open">
  <cn type="integer">0</cn>
  <pi/>
</interval>
```

```
<OMA>
  <OMS cd="interval1" name="interval_oo"/>
  <OMS cd="alg1" name="zero"/>
  <OMS cd="nums1" name="pi"/>
</OMA>
```

§ Heuristics. Many of Content MathML's function elements have more than one usage. They can take various numbers or types of arguments. OpenMath's `<OMS>`s do not tend to have alternative usages. To determine the proper OpenMath `<OMS>` of a Content MathML function, it is necessary to examine the arguments.

Example 1.6 Content MathML's `<int/>` is used for both definite and indefinite integration. It has two corresponding OpenMath `<OMS>` attributes: `defint` from the `calculus1` CD specifies a definite integral and `int` from the same CD is specifies an indefinite integral. To determine the correct translation to OpenMath, it is necessary to inspect the arguments of the function.

Heuristics can be used to deal with usage differences, as shown in the following example.

Example 1.7 Content MathML's `<partialdiff>` (below left) accepts bound variables as a list of indexes or series of actual variable names. The OpenMath equivalent of `<partialdiff>` (below right) does not recognize the variables as a series of variable names. To deal with this difference, we can assume x maps to index 1 and so on.

```
<apply>
  <partialdiff/>
  <bvar>
    <ci> x </ci>
  </bvar>
  <bvar>
    <ci> y </ci>
  </bvar>
  <apply>
    <ci> f </ci>
    <ci> x </ci>
    <ci> y </ci>
  </apply>
</apply>
```

```
<OMA>
  <OMS cd="calculus1" name="partialdiff"/>
  <OMA>
    <OMS cd="list1" name="list"/>
    <OMI> 1 </OMI>
    <OMI> 2 </OMI>
  </OMA>
  <OMA>
    <OMV name="f"/>
    <OMV name="x"/>
    <OMV name="y"/>
  </OMA>
</OMA>
```

§ External Semantics. If neither special transformations nor heuristics are sufficient to provide a faithful translation, one may use the Content MathML extension mechanism (Section 2.2) to refer to an OpenMath definition. To convert from OpenMath to Content MathML, a `definitionURL` attribute is constructed, using the `CD` and `name` attributes from the OpenMath `<OMS>` element, and added to the target Content MathML `<csymbol>` element. To do the reverse transformation, the `CD` and `name` attributes can be extracted from the `definitionURL` attribute.

Example 1.8 Content MathML does not have an equivalent of the definitions in the `bigfloat1` CD.

```
<apply>
  <csymbol encoding="OpenMath"
    definitionURL=
    "http://www.openmath.org/(no break)
     cd/bigfloat1.ocd#bigfloatprec"
  <ci> m </ci>
  <ci> r </ci>
  <ci> e </ci>
</apply>
```

```
<OMA>
  <OMS cd="bigfloat1" name="bigfloatprec"/>
  <OMV name="m"/>
  <OMV name="r"/>
  <OMV name="e"/>
</OMA>
```

```
<ci type="integer">n</ci>
```

```
<OMATTR>
  <OMATP>
    <OMS cd="mathmltypes" name="type"/>
    <OMS cd="mathmltypes" name="integer_type"/>
  </OMATP>
  <OMV name="n"/>
<OMATTR>
```

Figure 3.3. Content MathML (left) and OpenMath (right) giving the symbol n integer type

```
<cn type="rational">
  2
  <sep/>
  3
</cn>
```

```
<OMA>
  <OMS cd="nums1" name="rational"/>
  <OMI> 2 </OMI>
  <OMI> 3 </OMI>
</OMA>
```

Figure 3.4. Rational number by Content MathML (left) and OpenMath (right)

3.2 Examples of Mismatches. We now discuss how these strategies can be used in dealing with the differences between Content MathML and OpenMath. The examples also illustrate the strategies' limitations. Note that the strategies don't provide a one-to-one mapping between the two formats. Instead, they help preserve, as much as possible, semantics under translation. For some of the mismatches, none of the strategies are applicable. This situation is discussed in Section 4.

§ Type Information. In OpenMath any mathematical object, including compound objects, can be annotated with type information. Types are defined in CDs, and mathematical objects are annotated using `<OMATTR>` and `<OMATP>`, which are the "attribute constructor" and "attribute pair" elements. In Content MathML, type information can be specified directly only for numbers and identifiers, using the `type` attribute within the elements `<cn>` and `<ci>`, as shown in Figure 3.3. Although it would be possible to provide MathML annotations, using `<semantics>` and `<annotation>` or `<annotationXML>`, this does not place the information into Content MathML. The simple heuristic to apply in the translation of OpenMath type-attributed objects is: if the type-attributed OpenMath object is an integer or variable, then a special transformation can be applied; otherwise, employ the Content MathML extension mechanism.

§ Numbers with compound structure. OpenMath uses CDs to define numbers with compound structure, such as rational numbers (Figure 3.4). A number may be given in parts, which, in general, may themselves be compound objects. Content MathML, however, cannot represent numbers as a composition of compound mathematical objects. Rather, in Content MathML, a number must have components that are simple tokens given as text. The interpretation of the components is specified

```
<apply>
  <int/>
  <bvar>
    <ci> x </ci>
  </bvar>
  <apply>
    <sin/>
    <ci> x </ci>
  </apply>
</apply>
```

```
<OMA>
  <OMS cd="calculus1" name="int"/>
  <OMBIND>
    <OMBVAR> <OMV name="x"/> </OMBVAR>
    <OMA>
      <OMS cd="fns1" name="lambda"/>
      <OMA>
        <OMS cd="transc1" name="sin"/>
        <OMV name="x"/>
      </OMA>
    </OMA>
  </OMBIND>
</OMA>
```

Figure 3.5. $\int \sin(x)\, dx$ in Content MathML (left) and OpenMath (right)

by the `type` attribute, which must be one of `real`, `integer`, `rational`, `complex-cartesian`, `complex-polar` or `constant`. If, as shown in Figure 3.4, an OpenMath number can be represented in one of the forms allowed by Content MathML `<cn>`, a special transformation can be applied. Otherwise, external semantics may be used or a composite Content MathML may be constructed (*i.e.* as an expression, rather than a single `<cn>`).

§ Use of Lambda Bindings. In OpenMath, lambda bindings can be used to represent functions, as shown in Figure 3.5. For example, the `calculus1` CD operators `diff` and `int` can take a function whose arguments are specified with lambda bindings. In Content MathML, however, bound variables are not specified as part of the function, but rather as arguments to the operators. So, in translating from OpenMath to Content MathML, if a function expression does not give an explicit lambda binding, then there is no way to determine the bound variables. If the argument of such functions in OpenMath is not a lambda binding, we may use heuristics to guess the bound variable when translating from OpenMath to Content MathML. In the reverse translation, we need to reconstruct a lambda binding from the function application with the bound variables.

§ Miscellaneous Integer Functions. There is no direct Content MathML equivalent of OpenMath's `trunc` and `round` from the `rounding1` CD, and `abs` from the `arith1` CD. External references or special transformations constructing equivalent mathematical functions may be used.

We have seen that while the definitions of OpenMath and Content MathML allow many expressions to be translated precisely, there are several technical points on which the formats do not agree. Provided the application can be made to avoid these aspects, it is possible to have a natural translation between the corresponding concepts provided by the two standards. If a more robust solution is required, then another approach is needed, as described in the next section.

4 Conversion Using External Semantics

For a completely general solution for the translation between OpenMath and Content MathML it is necessary to forgo the translation between high-level concepts that sometimes do not match exactly. Instead, we must rely on mechanisms that give correct translation under all circumstances. In our approach we have relied on the general extension mechanisms of OpenMath and MathML. We therefore have a general, uniform way to convert between Content MathML and OpenMath.

In this setting, the conversion from Content MathML to OpenMath is similar to that described in Section 3. The only difference is that new CDs, precisely capturing the Content MathML semantics, may be used.

In contrast, the conversion from OpenMath to Content MathML is now quite different. We use a simple subset of Content MathML elements that correspond to the OpenMath expression-forming elements. These include `<ci>`, `<cn>`, `<csymbol>`, `<lambda>`, `<bvar>` and `<apply>`. Then *all* OpenMath symbols are translated to `<csymbol>` elements with references to the canonical OpenMath URLs. Effectively, we embed OpenMath in Content MathML syntax. In other words, no attempt is made to generate "native" Content MathML.

An example of such a conversion is given in Figure 4.6. The basic OpenMath constructs, such as the apply construct, `<OMA>`, are mapped to the corresponding Content MathML constructs. The symbol construct, `<OMS>`, maps to the MathML `<csymbol>` element, with an attribute containing the URL for the OpenMath semantics.

```
<math>
  <apply>
    <csymbol definitionURL=
     "http://www.openmath.org/(no break)
      cd/mathml/logic1#not"/>
    <apply>
      <csymbol definitionURL=
       "http://www.openmath.org/(no break)
        cd/mathml/logic1#and"/>
      <ci> A </ci>
      <ci> B </ci>
    </apply>
  </apply>
<math>
```

```
<OMOBJ>
  <OMA>
    <OMS cd="logic1" name="not"/>
    <OMA>
      <OMS cd="logic1" name="and"/>
      <OMV name="A"/>
      <OMV name="B"/>
    </OMA>
  </OMA>
</OMOBJ>
```

Figure 4.6. "Simplified" Content MathML (left) and OpenMath (right) markup of $\neg(A \wedge B)$

This strategy not only provides a solution to translating OpenMath elements for which no corresponding Content MathML construct exists, but also illustrates how OpenMath can be used to complement Content MathML's limited facilities for expressing the semantics of a wider range of mathematical subjects.

5 Implementation

We have implemented a translator based on the strategies discussed in Sections 3 and 4. The translator was implemented as a set of XSLT [8] stylesheets. The stylesheets are tested using `xt` [3], one of the implementations of XSLT. XSLT was chosen to implement the translator because it is designed specifically to transform XML expression trees. Another advantage is that many implementations of XSLT are free and widely available. The stylesheets are available at the website of our laboratory: http://www.orcca.on.ca/MathML.

There has been at least one other implementation of a Content MathML/ OpenMath translator [12]. This previous work discusses only the correspondence between the built-in semantics of the formats and is based in a REDUCE environment. In contrast, our open-ended conversion strategy does not restrict our attention to built-in semantics, and we have employed more widely available XML technology for our implementation.

It should be noted that our work is based on OpenMath version 1.0 and MathML 2.0. Since the development of our translator, both of the standards have evolved. At the time of writing, OpenMath 2.0 has been released and discussions for MathML 3.0 are underway. Although minor revisions will be made to the respective standards, our translation strategies can still be applied and the mismatches that we have identified here still exist.

6 Conclusion

OpenMath and Content MathML are two standards to encode the semantics of mathematical expressions. There are many similarities and differences between them.

We have described two translation strategies to convert between these two languages: The first strategy maps between the corresponding elements of the standards. It exposes a number of differences between OpenMath and Content MathML that must be dealt with. The second strategy uses the low-level structure of the formats to give precise embeddings of each within the other.

We have implemented these translation strategies using XSLT. The implementations have served two purposes: first as a proof of concept, to verify the validity of the approaches, and second, to fill a practical need, providing translator blocks in an architecture for mathematical web services [1].

Bibliography

[1] *MONET (Mathematics on the Net)*, http://monet.nag.co.uk/cocoon/monet/index.html.

[2] *The OpenMath Society*, http://www.openmath.org/.

[3] *XT*, http://www.jclark.com/xml/xt-old.html.

[4] R. Ausbrooks, S. Buswell, D. Carlisle, S. Dalmas, S. Devitt, A. Diaz, M. Froumentin, R. Hunter, P. Ion, M. Kohlhase, R. Miner, N. Poppelier, B. Smith, N. Soiffer, R. Sutor, and S. Watt, *Mathematical Markup Language (MathML) Version 2.0 (second edition), W3C Recommendation 21 October 2003*, World Wide Web Consortium (W3C), 2003, http://www.w3.org/TR/2003/REC-MathML2-20031021.

[5] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, and cois Yergeau Fran *Extensible Markup Language (XML) 1.0 (fourth edition) w3c Recommendation 16 August 2006*, 2006, http://www.w3.org/TR/2006/REC-xml-20060816.

[6] David Carlisle, *OpenMath, MathML, and XSL*, ACM SIGSAM Bulletin **34** (2000), 6–11.

[7] David Carlisle, James Davenport, Mike Dewar, Namhyun Hur, and William Naylor, *Conversion between MathML and OpenMath*, The OpenMath Consortium (2001).

[8] James Clark, *XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999*, World Wide Web Consortium (W3C), 1999, http://www.w3.org/TR/xslt.

[9] Stéphane Dalmas, Marc Gaëtano, and Stephen Watt, *An OpenMath 1.0 implementation*, Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC), 1997, pp. 241–248.

[10] Mike Dewar, *OpenMath: an overview*, ACM SIGSAM Bulletin **34** (2000), 2–5.

[11] W.N. Naylor and S.M. Watt, *On the relationship between OpenMath and MathML*, Electronic Proc. Internet Accessible Mathematical Communication (IAMC 2001), http://icm.mcs.kent.edu/research/iamc01proceedings.html.

[12] Luis Alvarez Sobreviela, *A Reduce-based OpenMath ↔ MathML translator*, ACM SIGSAM Bulletin **34** (2000), 31–32.