A Family of Modular XML Schemas for MathML

Stephen M. Watt   Yuzhen Xie

Ontario Research Center for Computer Algebra
Middlesex College Building
University of Western Ontario
London ON, N6A 5B7 Canada
http://www.orcca.on.ca

1. Introduction

XML Schema [1][2] provides a mechanism to specify a grammar to characterize properly formed XML documents for particular applications. They are more powerful than Document Type Definitions (DTDs), and their use is actively encouraged by the World Wide Web Consortium. We present here the first XML Schema for MathML.

MathML is a complex XML application that can, in fact, benefit from a schema definition. One problem in defining such a schema is to develop an architecture that captures the logical structure of MathML. The MathML definition provides two sorts of markup, *presentation markup* which captures the notational aspects of mathematics, and *content markup* which captures the meaning of mathematical expressions. Presentation and content MathML can be combined in certain well-defined ways, and capturing this structure in a schema is important.

Not all MathML applications will support the full MathML specification – for example, editing applications may support presentation MathML only, and mathematical computing applications might support content MathML only. It is therefore useful to provide XML Schemas for Presentation MathML and Content MathML independently. This is non-trivial as the definitions of the presentation and content element definitions must be decoupled for this purpose.

This paper describes a family of XML Schema modules which may be combined in various ways to provide, in an elegant manner, the first XML schema for full MathML as well as independent schemas for presentation and content MathML. The resulting schema family is available at http://www.orcca.on.ca/MathML/software.html.

2. Insufficiency of Schema Generation Tools

Two programs for automatically translating DTD into XML Schema exist, "dtd2xsd" (a Perl program) [4] and "dtd2xs" (a Java program) [5]. Their naïve idea is to use pattern matching to map meaningful DTD entities onto XML Schema constructs. Both claim to handle element and attribute specifications, simple types, attribute groups, and model groups.

We have found that the MathML 2.0 DTD is too complex for these tools to handle correctly. They might work for simple DTDs, as tested by the authors. However, for a complicated schema like MathML we have found problems, even errors, with the generated schemas. While we might imagine that future versions of these tools will be error-free, but that does not obviate the main difficulty: the generated schemas do not take advantage of whatever structure exists in the input DTD. The resulting schemas are consequently monolithic and very repetitive. (The schemas generated from the MathML 2.0 DTD were 230-300KB in size.) These are not suitable for hand maintenance and evolution and cannot be considered as replacement source for the DTD.

With this state of tool development, it appeared necessary to prepare the first XML Schemas for MathML by hand. This allows us to take advantage of structure, types, constraints, and use scenarios of MathML.

3. The Modular Structure

We defined eight modules in a hierarchical structure for the XML Schema for MathML. The hierarchical modular structure is illustrated in Figure 1.

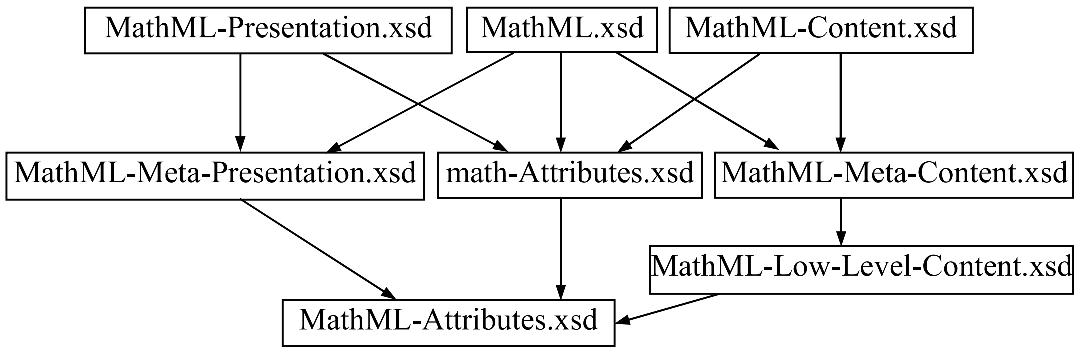

Figure 1 Hierarchical Modular Structure of XML Schema for MathML

The attributes shared by all the elements in MathML are defined in one module, called "MathML-Attributes.xsd" (2 KB). All the attributes accepted by the top-level element `math` are stated in "math-Attributes.xsd" (2KB).

The canonically empty content elements are declared in a separate module named "MathML-Low-Level-Content.xsd" (9 KB).

The “MathML-Meta-Presentation.xsd” (17 KB) defines a schema module for a pure presentation world. The presentation elements are defined without the possibility of having content elements as children. Consequently, only the presentation elements are allowed at the leaf level of the `<math>` element to form a schema for standalone presentation markup in “MathML-Presentation.xsd” (1 KB).

Likewise, the “MathML-Meta-Content.xsd” (19KB) defines schema for a pure content world. The content elements are defined without the possibility of having presentation elements as children. Only the content elements are allowed at the leaf level of `<math>` element to form a schema for standalone content markup in “MathML-Content.xsd” (1 KB).

In “MathML.xsd” (2 KB), the content models for presentation and content markup are redefined so that content elements are allowed to be present at the leaf level of presentation elements, and vice versa. A new type that includes both presentation elements and content elements is created for math element.

We see that this modular structure, as well as giving a much more flexible and versatile specification, is an order of magnitude smaller than a schema derived naively from the DTD.

4. Mechanisms Used

The eight modules have been created as individual schema files within a common XML namespace, “http://www.orcca.on.ca/MathML/Schema”. To allow their flexible reuse, all the elements and most of the attributes are declared globally in each module. To organize the constructs among these modules, the `<include>` and `<redefine>` mechanisms of the XML Schema language have been used.

The `<include>` element is used to bring into a schema the definitions and declarations contained in another document specified by the `schemaLocation` attribute. For example, the components in “MathML-Attributes.xsd” are added to the including schema “math-Attributes.xsd”, “MathML-Meta-Presentation.xsd”, and “MathML-Low-Level-Content.xsd” respectively by using an include statement. The elements, attributes and types in the included schema can be reused in the including schema. New groups and types can be defined or derived based on the existing ones. For example, in “MathML-Meta-Content.xsd” a number of element groups are defined according to the element declarations in “MathML-Low-Level-Content.xsd”.

In “MathML-Presentation.xsd”, the `<math>` element is defined with the type of `mathPresentationType`, which is constructed to contain the element group of `PresInCont` with the appropriate attribute lists. By this approach a partial schema for only presentation markup is formed. In the same way, a partial schema for only content markup is created in “MathML-Content.xsd”, with the type of math element defined to enclose `ContInPres` along with its attribute list.

The `<redefine>` mechanism is used to redefine simple and complex types, groups, and attribute groups that are obtained from external schema files specified in the `schemaLocation` attribute. Once the components are redefined they are taken to be part of the redefining schema's target namespace. In composing the whole XML Schema for MathML in "MathML.xsd", the group of `PresExpression` is redefined to enclose both the previously defined `PresExpression` group in "MathML-Meta-Presentation.xsd" and the `ContInPres` group defined in "MathML-Meta-Content.xsd". The group of `ContentExpression` is redefined to contain the previously defined `ContentExpression` group in "MathML-Meta-Content.xsd" and the `PresInCont` group defined in "MathML-Meta-Presentation.xsd". Concretely, we use the redefine mechanism as follows:

```
<redefine schemaLocation="http://www.orcca.on.ca/
          MathML/schemas/MathML-Meta-Presentation.xsd">
    <!-- redefinition of PresExpression group -->
    <group name="PresExpression">
        <choice minOccurs="0" maxOccurs="unbounded">
            <group ref="mms:PresExpression"/>
            <group ref="mms:ContInPres"/>
        </choice>
    </group>
 </redefine>

 <redefine schemaLocation="http://www.orcca.on.ca/
          MathML/schemas/MathML-Meta-Content.xsd">
    <!-- redefinition of ContentExpression group-->
    <group name="ContentExpression">
        <choice minOccurs="0" maxOccurs="unbounded">
            <group ref="mms:ContentExpression"/>
            <group ref="mms:PresInCont"/>
        </choice>
    </group>
 </redefine>
```

5. Concluding Remarks

We have created a family of XML Schema modules which allow a compact specification of content, presentation, and full MathML. These are an order of magnitude smaller than what is obtained from machine translation of the MathML DTD, and allow effective reuse and maintenance of components. We are now in a position to experiment with the use of XML schemas for MathML, to explore how they may be used to simplify and improve the design of various tools for MathML.

References

[1] W3C Recommendation: XML Schema Part 1: Structures, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/ May 2001.

[2] W3C Recommendation: XML Schema Part 2: Datatypes, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/ May 2001.

[3] W3C Recommendation: Document Type Definition (DTD), Extensible Markup Language (XML) 1.0, http://www.w3.org/TR 1998/REC-xml-19980210 February 1998
.
[4] Mary Holstege, et al, A Conversion Tool from DTD to XML Schema, http://www.w3.org/2000/04/schema_hack/, January 2001.

[5] J. Dudeck, et al, dtd2xs, http://puvogel.informatik.med.uni-giessen.de/dtd2xs/, March 2002.

[6] W3C Recommendation: Mathematical Markup Language (MathML) Version 2.0, http://www.w3.org/TR/2001/REC-MathML2-20010221