1. Introduction and Background

The purpose of this article is to show how Content MathML can be customized and extended. Together with an associated XSL stylesheet and processor, the customized MathML markup can be converted into standard Presentation MathML for display in a browser or for other purposes. Displayed documents with Presentation MathML can be further converted to PDF. Thus, in a sense, customized Content MathML can be viewed as an authoring platform for Presentation MathML.

Quite a bit of background information is necessary to understand this article: XML, MathML (in both its versions), and XSL. We will review these markup languages briefly in this introductory section, but it is well beyond the scope of this article to give exhaustive descriptions. Thus, this article will be most valuable to readers who have a basic knowledge of markup languages.

1.1 XML

All of the markup languages that we will discuss are members of the *eXtensible Markup Language* (XML) family. XML is a standard of the World Wide Web Consortium (W3C), the official standards body for web technologies. XML is perhaps best thought of as a meta-language for the creation of customized markup languages that are designed to encode specialized types of information and data. Particular XMLs of this type are often called *XML applications*. XML is rapidly becoming ubiquitous in the information technology world. A few examples of XML applications are

- XHTML (eXtensible HyperText Markup Language), the rigorous, XML version of HTML, which in turn is the basic language of the web.
- MathML (Mathematics Markup Language) in both flavors--Content MathML and Presentation MathML.
- XSL (eXtensible Style Language), a set of XML applications for formatting other XML applications and for transforming one XML application into another.
- SVG (Scalable Vector Graphics), an XML application for vector graphics.

- InkML InkML (Ink Markup Language), an XML application for “digital ink data” from a pen or stylus.
- XBRL (eXtensible Business Reporting Language), an XML application for financial information.

In addition to these large-scale, standard applications, there are many smaller and sometimes proprietary XML applications. For example, XML applications are often used to encode the parameters and data for various types of software, even Microsoft Office (OOXML). This article was written with the gedit text editor, to give another example, which uses a special XML applications for “snippet” libraries. As an example of a highly specialized XML application, you may want to view the XML file for the snippet library that I have customized for gedit. (Note however, that I did not have to write this XML file, or even know of its existence; gedit has a nice GUI interface for customizing snippet libraries.)

Moreover, XML applications can be combined to form *compound documents*. For mathematics teachers (and for this article), the compound document of most interest is XHTML (for basic expository text), embedded with MathML (for mathematical expressions), and perhaps also embedded with SVG (for graphics).

An XML document consists of tags that should be familiar to anyone who has written an HTML document. Tags can be either *binary* or *unary*, and can have *attributes*. Binary tags have an opening and closing part, with text (or other tags) inside. An example of a binary tag, from Content MathML is `<set></set>`. Inside this tag would be text and markup that would define a set. An example of a unary tag, from XHTML is `<img scr="dice.png" />`. This tag would insert an image into the document. The `src` attribute specifies the location of the image file. The rules for tags in XML are rigorous; a standards-compliant browser is forbidden to display a malformed XML file.

Structurally, an XML document is a rooted tree, a structure any mathematician can appreciate. The nodes of the tree are the tags and the children of a given node are the tags inside of that node.

Ideally, XML applications are “self-documenting”. The names given to the tags should meaningful, and the tree structure should make sense, at least in the context of the particular application. For example, without knowing anything at all about the underlying XML application (which happens to be Content MathML), you can probably guess the meaning of the following markup

```
<vector>
  <ci>x</ci>
  <ci>y</ci>
  <ci>z</ci>
</vector>
```

The price to be paid for the rigorous tree structure required by XML and for the self-documenting naming conventions is that XML documents tend to be extremely verbose, compared to other document models, such as LaTeX. Every bit of data in an XML document must be inside a binary tag, or in an attribute, or must be implicit in the tree structure. One often hears that XML is not intended to be written "by hand", but rather is to be generated by various software tools. Certainly everyone would agree that writing XML would be prohibitively tedious if every character in the markup had to be individually typed. On the other hand, I believe that sophisticated tools are not necessary. XML can be produced in reasonable time and with high accuracy with a simple text editor that has the following features:

- The ability to easily insert tags and larger chunks of boilerplate markup by means of a customizable "snippet library".
- Syntax checking
- The ability to expand and collapse the tree structure

One of the reasons that LaTeX has been so successful is that it can be written "by hand" with simple text editors that have some basic helpful features. Although WYSIWYG tools exist for LaTeX, few mathematicians that I know use them. The basic procedure is write, compile, and view. So should it be with XHTML + MathML.

The XML applications that we need to understand for this article are MathML (both Presentation and Content), and XSL. For more general information, visit the XML site at W3C.

1.2 MathML

The *Mathematics Markup Language* (MathML) is a language for the encoding of mathematical expressions. MathML 1.0 was originally released as a recommendation by the W3C in 1998. MathML 2.0 was released as a recommendation in 2003, and work is currently underway on the latest version, MathML 3.0. The grand vision is for MathML to be a modern, general purpose language that could be used to encode mathematics for

- web browsers
- typesetting systems
- computer algebra systems
- voice synthesis systems

and many other applications. With a single, standard language, such systems could communicate mathematics seamlessly to users and to one another. Thus, a mathematical expression in MathML could be displayed properly in a web browser, printed properly on paper, spoken properly in a voice synthesis system, and operated upon in a computer algebra system.

MathML actually consists of two separate languages, *Presentation MathML* and *Content MathML* (both XML applications, of course). Presentation MathML primarily encodes the two-dimensional, left-to-right layout of mathematical notation (individual symbols, parentheses and other fences, sub and superscripts, etc.). Content MathML, on the other hand, encodes primarily the structure and semantics of a mathematical expression (sums, products, integrals, derivatives, etc.). In fact however, the distinction between the two is not as clean as one might think--it's actually quite hard to separate the meaning of mathematics from its notation. Moreover, as we will see, even Presentation MathML encodes much of the structure of a mathematical expression. We will study the two versions of MathML in more detail in Section 2.

1.5 XSL

The *eXtensible Style Language* (XSL), is a set of XML applications for specifying the style of XML documents, and more importantly, for transforming one XML language into another. In the later case, the XML application is referred to as XSLT, for *XSL Transformations*. Thus, the basic model for XSLT is

- a source XML document
- an XSLT stylesheet that defines the transformations
- a processor that applies the XSLT stylesheet to the source XML document to produce the output XML document

There are many XSLT processors available, both proprietary and open source, both commercial and free. The Firefox browser has a built in XSLT processor that works dynamically. A very nice free add-on to Firefox, named XSL Results, allows you to capture the output document. In addition, XSLT processors are available for integrated design environments (IDEs) such as Eclipse and Netbeans.

XSLT is a *template language*. The processor searches the tree structure of the source XML document looking for patterns that match a template. When a match is found, the transformations are applied. Also, XSLT has the logical structures of a programming language (with `if-then`, `for`, and `switch` constructions, for example). Thus, XSLT is the most complicated of the markup languages considered in this article. As with the other languages, we will study XSLT in this article primarily through examples. For detailed information, go to the XSL site at W3C

In particular, XSLT can be used to transform Content MathML into Presentation MathML, which of course is our main example of interest. The general model above becomes

- an XHTML document with embedded Content MathML

- an XSLT stylesheet that defines how to map Content MathML into Presentation MathML
- a processor that applies the XSLT stylesheet to the source document to produce an XHTML document with embedded Presentation MathML.

An XSLT stylesheet, ctop.xsl, for transforming Content MathML into Presentation MathML has been written by David Carlisle and is posted on the W3C MathML site. Most authors who write Content MathML use this stylesheet to display the MathML in Firefox. Customizing and extending this stylesheet is the subject of the Section 3.

1.6 Browser Support

This article is primarily concerned with the (now decade-old) problem of how to get mathematics in web pages. Thus, browser support for MathML is a critical issue.

Firefox

The Firefox browser has the best support for MathML. Firefox has native support for Presentation MathML, and supports XSLT. This implies, and is much stronger than, the statement that Firefox supports Content MathML. Thus, a webpage with Content MathML has a reference to an XSLT stylesheet (typically ctop.xsl). When processing the page, Firefox first applies the stylesheet to translate the embedded Content MathML into Presentation MathML, which it then renders. The fact that Firefox supports XSLT means that Content MathML can be customized. Indeed, one could wire a completely new XML application for mathematics by simply writing the XSLT stylesheet that translates the new language into Presentation MathML. Native support for Presentation MathML and support for XSLT is the ideal configuration. Hopefully, someday, all browsers will take the same approach. Finally, Firefox is available for the Windows, Linux, and Mac operating systems.

Internet Explorer

Internet Explorer supports MathML on Windows, via the MathPlayer plug-in. MathPlayer supports Presentation MathML and the standard version of Content MathML. Evidently however, MathPlayer does not have an XSLT processor, but rather renders Content MathML in a fixed way. This, of course, defeats one of the main reasons for using Content MathML, since the author has no control over the style and presentation of Content MathML elements. The lack of support for XSLT also means that customized and extended versions of Content MathML will not work in MathPlayer directly. Rather, the extended Content MathML must first be translated, using an external XSLT processor, into Presentation MathML before being published.

Other Browsers

The Opera Browser has limited support for Presentation MathML using the *Cascading Style Sheet* (CSS) style language. Opera supports XSLT generally, but seems unable to render Content MathML via an XSLT stylesheet. The Safari and Chrome browsers have no support for Presentation MathML, to the best of my knowledge.
