MathML without Plugins using VML

CS297 Final Report

Namon Nuttayasakul
namonpick@hotmail.com

Advisor: Dr. Chris Pollett

1. Introduction

The standard way to view math on the web is to write the mathematical document using LaTeX and use a conversion program such as latex2html [D96], WebEQ [D02], TtH [H01], or HeVeA [M02]. The first two of these programs generate graphic images (WebEQ can output MathML and applet code) for each mathematical formula on the page. This makes such web-documents both slow to load and hard to maintain as they now consist of many files, one for each formula. The latter two programs convert LaTeX directly into a single HTML file and try to draw the equations as best as possible given the limitations of HTML. These programs are very fast and often produce reasonable results. However, picture environments and the like still must be output using graphics images. MathML is an XML based mark-up language for displaying math on the web and there exist programs to convert both TeX and LaTeX to this format. Unfortunately, neither Netscape nor Internet Explorer natively supports this language. SVG [W3C01] and VML [W3C98b] are XML-defined vector mark-up languages in which both math notations and picture environments could easily be rendered. Currently, VML is natively supported in Internet Explorer but, SVG is neither natively supported by Internet Explorer nor Netscape.

Style-sheet transformations are rules which are applied by the browser or by the server to a tag when it is read or before it is transmitted. They basically provide a mechanism by which a document can be "compiled" into a format displayable by a browser. Netscape currently supports CSS2 (cascading style sheets level 2) [W3C98a] with the intention to have built in support for XSLT (eXtensible Stylesheet Language Transformations) [W3C99], the latter being stronger and more flexible. Internet Explorer supports XSLT. CSS2 has some support for tag-replacement by hypertext. This allows tags to be replaced by ECMAscript (aka Javascript) code [ECMA99] which can in theory output SVG code.

This proposed project uses the XSLT transformation language to convert XML files that contains MathML to XML file containing HTML and VML. The document only requires a single processing instruction linking to the stylesheet. The stylesheet transforms the supplied XML file, adding whatever markup is required to render MathML in the current browser, and pass the resulting document to the browser for rendering. Clearly this does require that the browser supports XSLT transformations, which does mean that a relatively new browser is required, however the current versions of (at least) Internet Explorer supports XSLT, so while XSLT support is not universal it is, or soon will be, available on the majority of desktop browsers. Thus, in both Netscape and Internet Explorer clients it should be feasible produce style-sheet transformations from MathML to the target language. Nevertheless, such a translation would be difficult to perform and is worthy of a masters.

This document is organized as follows. Section 2 states the goal of this master project in detail. Section 3 provides information on the related research and background of this project. Section 4 covers each deliverable and what I have accomplished in this semester in detail. Section 5 contains conclusions of my research completed so far.

2. Master Project's Goal

This section discusses the scope of the work in CS298 and the final result of the Master Project. Since, there are many transformation programs that convert from LaTeX to MathML currently available. With all the considerations of the time and the large scope of the LaTeX language, I think it is not necessary to develop a stylesheet transformation to convert LaTeX directly into VML. The LaTeX user can just use one of the LaTeX – MathML conversion program and then apply this project's XSLT transformation to render MathML in the Web.

For MathML user, there are two ways of encoding mathematical data using MathML: Content Markup or Presentation Markup. Content markup is concerned with the semantics of mathematics. Presentation markup is concerned with the rendering of mathematics. Currently, there are also several software applications that helps generate MathML easily with powerful Graphic User Interface. Most of these programs generate the MathML in presentation mode, which allows the user to display most of the mathematical expression. However, there is a MathML Content2Presentation Transformation (MathMLc2p), written in XSLT, is able to translate content markup expressions into presentation markup expressions automatically.

Within the three months period in CS298 Master Writing Project Course, the goal of this master's project is to develop a stylesheet transformation from MathML in presentation mode with default rendering and some important attributes into VML. In particular, all the necessary mathematical rendering transformation will be developed; however, some additional attributes that are used to enhance the rendering such as the spacing attributes, or the ability to split an expression if it is too long will be ignored, since this kind of rendering is software/device dependent.

The list of MathML tags that will be translate in the Master Project is provided below:

Category	Tag	Description
Token Elements:	<mi>	identifier
	<mn>	number
	<mo>	operator, fence, or separator
	<mtext>	text
	<mspace/>	space
	<ms>	string literal
General Layout:	<mrow>	group any number of sub-expressions horizontally
	<mfrac>	form a fraction from two sub-expressions
	<msqrt>	form a square root sign (radical without an index)
	<mroot>	form a radical with specified index
	<mstyle>	style change
	<merror>	enclose a syntax error message from a preprocessor
	<mpadded>	adjust space around content

	<mphantom>	make content invisible but preserve its size
	<mfenced>	surround content with a pair of fences
	<mrow>	group any number of sub-expressions horizontally
	<mfrac>	form a fraction from two sub-expressions
	<msqrt>	form a square root sign (radical without an index)
Scripts and Limits:	<msub>	attach a subscript to a base
	<msubsup>	attach a subscript-superscript pair to a base
	<munder>	attach an underscript to a base
	<mover>	attach an overscript to a base
	<munderover>	attach an underscript-overscript pair to a base
	<mmultiscripts>	attach prescripts and tensor indices to a base
Tables:		<mtable>
		<mtr>
		<mtd>
		<maligngroup/>
		<malignmark/>
Actions:	<maction>	bind actions to a sub-expression

Another intention of this project is to process the translation on the client-side. However, if the translation is done client-side then one is essentially sending the compiler along with the document, which makes it hard to sell this kind of product. To handle this issue, we will investigate ways to make the stylesheets sent only applicable to the given document requested.

3. Related Work and Background

This section will introduce related work and also the other technologies available to view MathML on the Web.

MathML is becoming more and more widely used. Though, at the current time few browsers have "native" support for MathML, and only one has native support for the Content part of MathML. Currently, Mozilla version 0.9.9 is the only browser that fully supports MathML in both presentation and content mode. The Amaya browser supports only the presentation mode of MathML. However, popular browsers such as Internet Explorer (IE) or Netscape still need some plug-ins or a range of extensions that will render MathML (in particular, WebEQ and MathPlayer from Design Science and Techexplorer from IBM) in order to make them able to display MathML. Below is the summary list of browsers that will render the pages categorized by the operating system as shown:

Operating System	Browser with necessary configuration
Windows	IE 5.0 with the Techexplorer plug-in
	IE 5.5 with either the MathPlayer or Techexplorer plug-ins
	IE 6.0, optionally with MathPlayer or Techexplorer plug-ins
	Netscape 6.1 with Techexplorer plug-in
	Amaya (Presentation MathML only)
	Mozilla 0.9.9
Macintosh	IE 5.0 with Techexplorer plug-in
	Mozilla 0.9.9
Linux/Unix	Netscape 6.1 with Techexplorer plug-in
	Mozilla 0.9.9
	Amaya (Presentation MathML only)

The disadvantage of using a third party extension to render MathML within a web page is that it requires specific markup to specify the rendering extension. The use of such markup ties the document to one particular platform; whereas, the ideal of publishing information on the Web is that it should be accessible to all using a range of tools.

Related work that contributes to make MathML viewable on the Web is a stylesheet [W3C22a] provided by W3C for XHTML file that has MathML code as an embedded tag. This is a mechanism uses the XSLT transformation language to allow XML file that contains MathML to convert to XML file containing XHTML. This will work in most cases (but not on Internet Explorer: for security reasons IE will not execute an XSLT stylesheet that is not located on the same server as the XHTML+MathML document. However, there are alternatives such as processing it on the client side. This mechanism basically transforms MathML document into an XHTML document, which of course will give a result that is not as satisfying as it tries to draw the mathematical expression using XHTML capability. The result will definitely be of lower quality in compared to that proposed by this master's project which uses a Vector graphic language to draw the each expression. Furthermore, my project will also give pages with better printing quality.

4. Deliverables Organization for CS297

In CS297 course, I have finished five deliverables each of which contribute to the learning process for the final project. The five deliverables are listed as follow:

1. Become reasonably proficient with VML and SVG. This will be demonstrated by creating an image of a sunset in both these languages.

The purpose of this deliverable was to learn and experiment with the currently available vector markup languages. I need to become proficient with some of the functions and features of the language in order to be able to render mathematical expression as vector graphics in the final project.

2. Become reasonably proficient at MathML and LaTeX. This was to be demonstrated by reproducing in these languages pages 130 and 131 of A Guide to LaTeX2e.

The purpose of this deliverable was to study the syntax and the results of most of the tags in LaTeX and MathML as I need to keep that in mind and try create the comparable or better results in the final stage.

3. Create a DTD for drawing matrices. Write a XSL transform to render this language in VML.

The purpose of this deliverable was very important: to learn the structure of XML markup languages by creating my own DTD for drawing matrices. This will help me fully understand the structure of the MathML and LaTeX languages. In this deliverable, I also started to explore on the way to make a transformation from XML file to HTML file with VML embeded tags using XSL style sheet transformation language.

4. Get the MathML matrix related, apply, minus, times, divide, and eq tags to translate to VML and to SVG via XSLT.

The purpose of this deliverable was to show that I am ready and have all the necessary knowledge to be able to produce the transformation of some tags in MathML to HTML file with VML embedded tags. This is the beginning stage of the real project I am going to deliver in the next semester. In this deliverable, I get a chance to fully explore the stylesheet transformation language and the results it produces. I took notes on good and bad coding techniques I experienced in this stage as ways to improve results and the coding style in the final project.

5. A first semester report on project of length between 10-20 pages. This final report is the result of the fifth deliverable.

The following sections describes each deliverable in details.

4.1 Deliverable 1

Deliverable Date: 02/08/02

Deliverable: To create an image of a sunset in VML and SVG.

Purpose: The purpose of this deliverable was to learn and experiment with the currently available vector markup languages. I needed to become proficient with some of the functions and features of the language in order to be able to render mathematical expression in the vector graphic for the final project.

Description: Today, the most available, well-known vector markup languages are VML and SVG. According to my research during this semester, there are many advantages of vector graphic format over other image formats, particularly over JPEG and GIF, the most common graphic formats used on the Web today. Advantages of vector graphic format over other graphic formats are provided below:

- **Plain text format**: Vector graphic files can be read and modified by a range of tools, and are usually much smaller and more compressible than comparable JPEG or GIF images.
- **Scalable**: Unlike bitmapped GIF and JPEG formats, vector format images can be printed with high quality at any resolution, without the "staircase" effects you see when printing bitmapped images.
- **Zoomable**: Vector graphic allow you to zoom in on any portion of an image and not see any degradation.
- **Searchable and selectable text**: Unlike in bitmapped images, text in vector graphic markup language is selectable and searchable. For example, you can search for specific text strings, like city names in a map.
- **Scripting and animation**: Vector graphic markup format enables dynamic and interactive graphics far more sophisticated than bitmapped or even Flash™ images.
- **True XML**: As an XML language, vector graphic markup language such as VML and SVG offer all the advantages of XML:
    - Interoperability
    - Internationalization (Unicode support)
    - Wide tool support
    - Easy manipulation through standard APIs, such as the Document Object Model (DOM) API
    - Easy transformation through XML Stylesheet Language Transformation (XSLT).

The following sections will briefly describe VML and SVG, then gather some reasons for choosing VML for this Master's project.

VML

Vector Markup Language (VML) is an XML, text-based markup language. VML has specifications that include features allowing applications to store higher-level application-specific private data within the graphics file. These features promote a higher-level interchange of graphics between applications. It also provides new ways of combining scripting with the Document Object Model (DOM) API to control a Web page's graphical elements. It provides an easy, standard way for a script writer to manipulate the graphic without requiring the use of special software tools.
VML is supported by Internet Explorer version 5 or greater. Many software applications, such as Microsoft Word, Excel, and Power Point can automatically convert pictures in vector graphic format by outputting XML file containing VML.

SVG

Scalable Vector Graphics (SVG) is a new graphics file format and Web development language based on XML. SVG enables Web developers and designers to create dynamically generated, high-quality graphics from real-time data with precise structural and visual control. With this powerful new technology, SVG developers can create a new generation of Web applications based on data-driven, interactive, and personalized graphics. SVG was created by the World Wide Web Consortium (W3C), and now over twenty organizations, including Sun Microsystems, Adobe, Apple, IBM, and Kodak, have been involved in defining SVG.

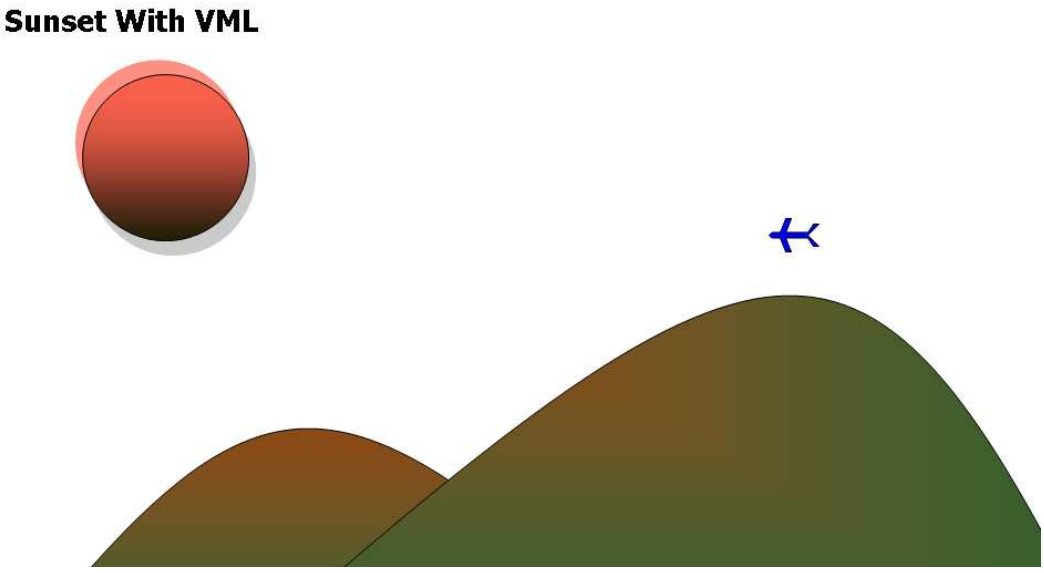
Reasons for chosen VML for my Master's project

The main reasons for choosing VML as my target language is that VML is now natively supported by Internet Explorer 5.0 or above. This makes it possible to display MathML by using stylesheet transformation to render it in VML without using any plugins. Even though, SVG is an international standard language, it is not yet supported by any of the popular browsers. There are several tools available for displaying MathML through SVG, but to be able to render it in popular browsers, the user still needs some kinds of Plugins.

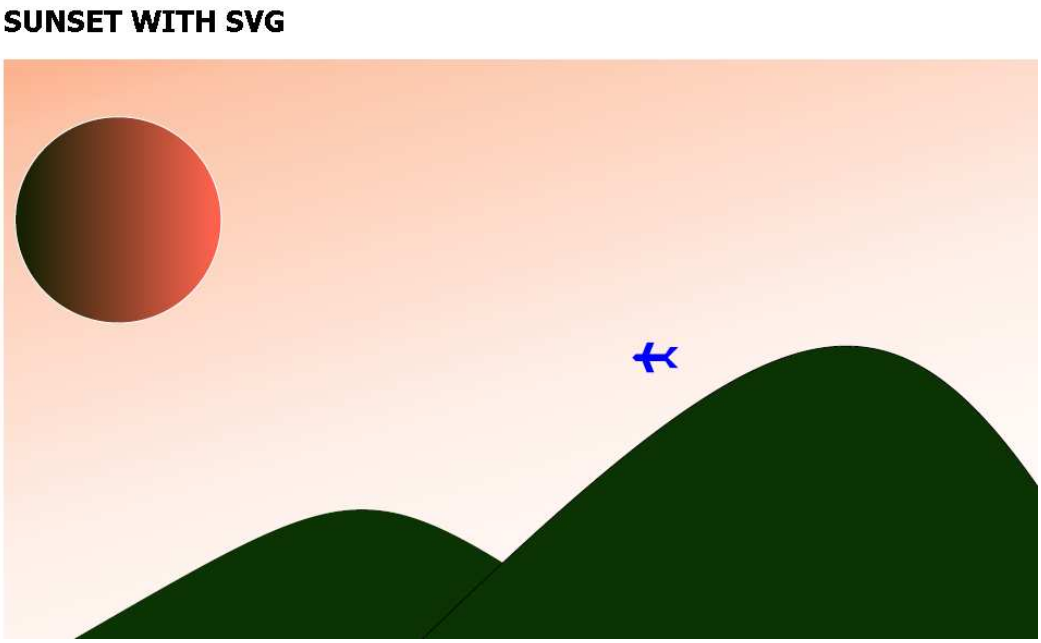
It is true that SVG seems to be better, cleaner, and a more complete standard, but considering the purpose of this project which only needs the use of the vector graphic to draw mathematical expression, which VML's capabilities are more than sufficient to accomplish this project.

Example Result:

VML's example result:



SVG's example result:



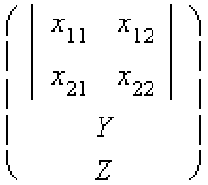
4.2 Deliverable 2

Deliverable Date: 03/01/02

Deliverable: To reproduce by hand in pages 130 and 131 of “A Guide to LaTeX2e” in LaTeX and in MathML.

Purpose: The purpose of this deliverable was to gain experience with the most common tags of LaTeX and MathML and know what they output in common translating mechanisms as I need to keep that in mind and try to create the comparable or better results in the final stage.

Description: I wrote a LaTeX document and MathML document for the pages 130 and 131 of “A Guide to LaTeX2e” book. Looking at the results, LaTeX documents that are viewed by DVI viewer gave a better rendering result compared to the MathML document viewed by the Amaya browser. With the Amaya browser, some of the MathML elements give an unsatisfying rendering such as matrix brackets that has unconnected line as shown below.



MathML

MathML is the standard XML application used for displaying mathematical notation and content on the web. The goal of MathML is to enable mathematics to be served, received, and processed on the Web, just as HTML has for text. MathML provides the ability to control the presentation and the meaning of such expressions. It does this by providing two sets of markup tags, one set presents the notation of mathematical data in markup format (presentation mode), and the other set relays the semantic meaning of mathematical expressions (content mode), enabling complex mathematical and scientific notation to be encoded in an explicit way

LaTeX2e

The history of LaTeX2e begins long ago with the program called TeX. TeX is a computer program for typesetting documents, created by D. E. Knuth. It takes a suitably prepared computer file and converts it to a form that may be printed on many kinds of printers, including dot-matrix printers, laser printers and high-resolution typesetting machines.

LaTeX, written by L. B. Lamport, is one of a number of variation of TeX. It is particularly suited to the production of long articles and books, since it has facilities for the automatic numbering of chapters, sections, theorems, equations etc., and also has facilities for cross-referencing. LaTeX2e is a new version of LaTeX and is a standard program for producing mathematical documents.

Example Result

Example result in MathML:

$$\sum_{p_1 < p_2 < \ldots < p_{n-k}}^{(1,\,2,\,\ldots,\,n)} \Delta \begin{matrix} p_1 & p_2 & \cdots & p_{n-k} \\ p_1 & p_2 & \cdots & p_{n-k} \end{matrix} \sum_{q_1 < q_2 < \ldots < q_k} \begin{vmatrix} a_{q_1 q_1} & a_{q_1 q_2} & \ldots & a_{q_1 q_1} \\ a_{q_1 q_1} & a_{q_1 q_2} & \ldots & a_{q_1 q_1} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ a_{q_1 q_1} & a_{q_1 q_2} & \ldots & a_{q_1 q_1} \end{vmatrix}$$

Example result in LaTeX2e:

$$\sum_{p_1<p_2<\cdots<p_{n-k}}^{(1,2,\ldots,n)} \Delta \begin{matrix} p_1 p_2 \cdots p_{n-k} \\ p_1 p_2 \cdots p_{n-k} \end{matrix} \sum_{q_1<q_2<\cdots q_k} \begin{vmatrix} a_{q_1q_1} & a_{q_1q_2} & \cdots & a_{q_1q_k} \\ a_{q_2q_1} & a_{q_2q_2} & \cdots & a_{q_2q_k} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ a_{q_kq_1} & a_{q_kq_2} & \cdots & a_{q_kq_k} \end{vmatrix}$$

4.3 Deliverable 3

Deliverable Date: 04/16/02

Deliverable: Create a DTD for drawing matrices. Write a XSL transform to render this language in VML.

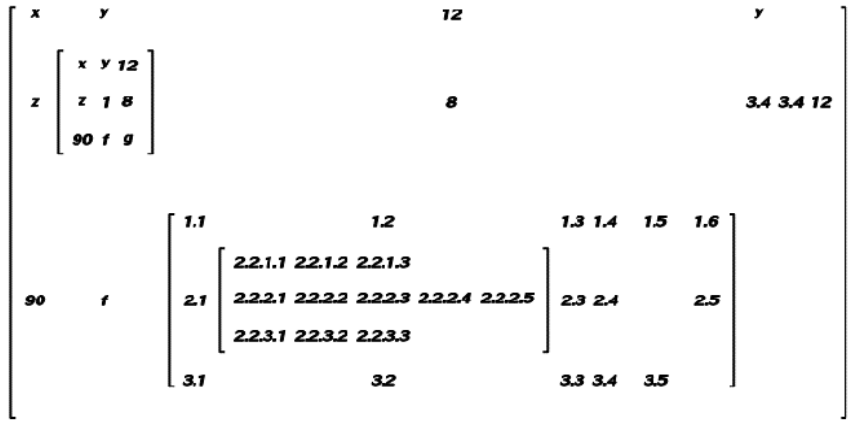
Purpose: The purpose of this deliverable was to learn the structure of the markup language by creating my own DTD for drawing matrices. This will help me fully understand the structure of the MathML and LaTeX languages. In this deliverable, I also started to explore ways to make a transformation from XML to HTML with VML embedded tags using XSL style sheet transformation language.

Description: For this deliverable a Data Type Definition (DTD) was given for matrices and an example document is written in this language. An XSLT document is then used to translate this particular XML document to VML document for viewing.

The main reason for using the author's own DTD file was to reduce the complexity in the XSLT transformation at this beginning stage of the project. Eventually, MathML document will be translate to VML using XSLT.

One of the coding difficulties in this deliverable was to allow each row in the matrix to have different number of column. This problem exists in presentation mode only, since it allow each row in the same matrix to have any numbers of columns. On the other hand, content mode in MathML has stricter rules which will not allow this type of issue. From my experience with this deliverable, I discovered that implementing stylesheet transformation from the presentation mode in MathML is more difficult than that of the content mode. Since, presentation mode does not have any rules to control the semantic meaning of the mathematical equation, it allow any kind of rendering even though it does not make any sense in the real mathematical environment. This makes the transformation harder because it has to deal with all the possible cases of rendering. It is obvious that if MathML presentation mode stylesheet transformation is successfully accomplished, the content mode transformation can be implemented with ease.

Example Result
Below is the example result of a nested matrix in MathML presentation mode:



The Technical Specification for the author's matrixDTD:

1) The "matrices" element

Description:
This element is the root of the document.

XML template:
<matrices></matrices>

2) The "matrix" element
Description:
A matrix element can cantain many nested matrices inside itself. It can also contain many rows elements. This element has two attributes which are "numberOfRows" and "numberOfColumns"

```
<!ATTLIST matrix
        numberOfRows        CDATA #REQUIRED
        numberOfColumns CDATA #REQUIRED
>
```

Attribute Descriptions:
numberOfRows is used for setting the number of rows in this matrix.
numberOfColumns is used for setting the number of columns in this matrix.

3) The "row" element
Description:
This element is used to define each row in the matrix. The row element can contain many columns elements.

4) The "column" element
Description:
This element is used define each column within this row in the matrix. This element has one attribute. Column can also contain nested matrices.

```
<!ATTLIST column
        content CDATA #REQUIRED
>
```

Attribute Description
The content attribute is used to hold the element to store in the column of the parent's row in the matrix.

4.4 Deliverable 4

Date: 04/03/02

Deliverable: Get the MathML matrix related, apply, minus, times, divide, and eq tags to translate to VML and to SVG via XSLT.

Purpose: The purpose of this deliverable was to show that I am ready and have all the necessary knowledge to be able to produce the transformation of some MathML tags to HTML file with embedded VML. This is the beginning stage of the real project I am going to deliver in next semester. In this deliverable, I get a chance to fully explore the stylesheet transformation language and the results it produces. I took notes on good and bad coding techniques I experienced in this stage as a way to improve the results and the coding style in the final project.

Description: This deliverable gives a stylesheet transformation to VML for some particular tags in MathML language such as: matrix related tags, apply, minus, times, divide, and eq. It differs from the last deliverable in that it translates those tags from the content markup mode, unlike last deliverable which transform according to the presentation markup mode.

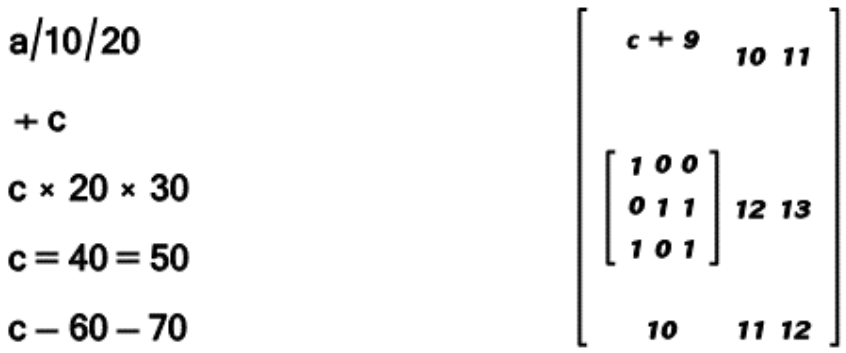
The use of content markup rather than presentation markup for mathematics is sometimes referred to as semantic tagging. The parse-tree of a valid element structure using MathML content elements corresponds directly to the expression tree of the underlying mathematical expression. However, even in such simple expressions as X + Y, some additional information may be required for applications such as computer algebra. Are X and Y integers, or functions, etc.? For example, do we have X+Y or just +X. The interpretation must determine which operand a given operator should be applied to. This additional information is referred to as semantic mapping. In MathML, this mapping is provided by the semantics, annotation and annotation-xml elements.

The semantic elements are the container element for a MathML expression together with its semantic mappings. Semantic elements expect a variable number of child elements. The first is the element (which may itself be a complex element structure) for which this additional semantic information is being defined. The second and subsequent children, if any, are instances of the elements annotation and/or annotation-xml.

Our stylesheet transformation interprets the MathML expression according to some features in the content markup mode explained above such as, the ability to determine which operand should the operator apply to (eg. A-B or -B)

Example Result:

Below is an example result of the basic operators' and matrix's rendering in MathML content mode:



4.4.1 Technical Information:

1) matrix and matrixrow
The matrix element is used to represent mathematical matrices It has zero or more child elements, all of which are matrixrow elements. These in turn expect zero or more child elements that evaluate to algebraic expressions or numbers. These sub-elements are often numbers, or symbols as in the example below:

```
<matrix>
  <matrixrow> <cn> 1 </cn> <cn> 2 </cn> </matrixrow>
  <matrixrow> <cn> 3 </cn> <cn> 4 </cn> </matrixrow>
</matrix>
```

The matrixrow elements must always be contained inside of a matrix, and all rows in a given matrix must have the same number of elements. Note that the behavior of the matrix and matrixrow elements is substantially different from the mtable and mtr presentation elements.

2) apply
Its purpose is to apply a function or operator to its arguments to produce an expression representing an element of the codomain of the function. It is involved in everything from forming sums such as a + b as in

```
<apply>
  <plus/>
  <ci> a </ci>
  <ci> b </ci>
</apply>
```

through to using the sine function to construct sin(a) as in

```
<apply>
  <sin/>
  <ci> a </ci>
</apply>
```

or constructing integrals. Its usage in any particular setting is determined largely by the properties of the function (the first child element). However, in this deliverable the sin and integrals transformations are not provided.

3) Subtraction (minus)
The minus element is the subtraction operator.
The minus element can be used as a unary arithmetic operator (e.g. to represent - x), or as a binary arithmetic operator (e.g. to represent x- y).

```
<apply> <minus/>
  <ci> x </ci>
  <ci> y </ci>
</apply>
```

4) Addition (plus)
The plus element is the addition operator (e.g. to represent x + y).

```
<apply>
  <plus/>
  <ci> x </ci>
  <ci> y </ci>
  <ci> z </ci>
</apply>
```

5) Multiplication (times)
The times element is the multiplication operator (e.g. to represent x * y).

Example

```
<apply>
  <times/>
  <ci> a </ci>
  <ci> b </ci>
</apply>
```

6) Equals (eq)
The eq element is the relational operator `equals'(e.g. to represent x = y).

Example

```
<apply>
  <eq/>
  <ci> a </ci>
  <ci> b </ci>
</apply>
```

7) Division (divide)
The divide element is the division operator (e.g. to represent x / y).

```
<apply>
  <divide/>
  <ci> a </ci>
  <ci> b </ci>
</apply>
```

4.4.2. Example High-level design of Deliverable 4 stylesheet transformation

```
<xsl:stylesheet>

<!-- Declare variables -->

<!-- match root (<math>tag) -->
<xsl:template match="/">
  <html>
    <!-- header in vml -->
    <!-- Declare VML Shapes for each mathematical symbols -->
  </html>
</xsl:template>

<!-- If match "matrix", add table header and set align to "center" -->
<xsl:template match="matrix">
   <!-- sets table width and align at center and Add each row -->
   <xsl:apply-templates/>
</xsl:template>

<xsl:template match="matrixrow[position() = 1 ]">
 <!-- if row is the first one then add bracket into it -->
 <!-- The left and right brackets are in the first row of table (in HTML). Therefore,
the rowspan must be the number of rows of the entire matrix -->
</xsl:template>

<xsl:template match="matrixrow[position() &gt; 1]">
 <!-- if row is not the fist one then just add new row of table -->
</xsl:template>

<xsl:template match="cn">
  <xsl:if test="parent::matrixrow">
     <!-- add number in the matrix's cell -->
   </xsl:if>
</xsl:template>

<xsl:template match="ci">
 <xsl:if test="parent::matrixrow">
         <!-- add variable in the matrix's cell -->
 </xsl:if>
</xsl:template>

<!-- Match Apply tag -->
<xsl:template match="apply">
  <xsl:apply-templates/>
</xsl:template>
```

```
<xsl:template match="eq">
        <xsl:call-template name="displayOperator">
          <xsl:with-param name="operator" select="'#equal'"/>
        </xsl:call-template>
</xsl:template>

<xsl:template match="minus">
        <xsl:call-template name="displayOperator">
          <xsl:with-param name="operator" select="'#minus'"/>
        </xsl:call-template>
</xsl:template>

<xsl:template match="plus">
        <xsl:call-template name="displayOperator">
          <xsl:with-param name="operator" select="'#plus'"/>
        </xsl:call-template>
</xsl:template>

<xsl:template match="times">
        <xsl:call-template name="displayOperator">
          <xsl:with-param name="operator" select="'#times'"/>
        </xsl:call-template>
</xsl:template>

<xsl:template name="displayOperator">
    <!-- display operator according to the name of template that called -->
</xsl:template>

</xsl:stylesheet>
```

5. Conclusion

The desire to display math on the Web has been around for more than a decade. A number of mechanisms previously proposed are still not adequate. This project proposes a breakthrough technique to display math on the Web without any hassle by transforming XML file containing MathML to XML file containing HTML and VML using XSLT stylesheet transformation. Using this approach, Mathematical expressions can be render on the Web without using any type of Plugins. Furthermore, it takes shorter time to view the result than the conventional way that use image file such as GIF or JPEG and also gives a better rendering result than those displaying the result using plain HTML.

During this CS297 course, I have learned and gained experience with several languages involving in viewing mathematical expression on the Web, such as, VML, SVG, LaTeX, XML, MathML, and XSLT throughout all four deliverables. In CS298, I will deliver the complete version of a stylesheet transformation from presentation mode MathML mode with default rendering and some important attributes into HTML and VML. Completing the entire conversion from presentation mode MathML to HTML and VML is not an easy task. However, the approach that I will use in CS298 will be similar to the transformation I have done in Deliverable 4 (Deliverable 4's high level design is shown in section 4.4.2.) Lastly, the translation will be done on the client-side with a modified approach to avoid sending the entire stylesheet transformation code to the end user.

Bibliography:

Van Ossenbruggen, J., Hardman, L. Rutledge, L., and Eliëns, A. "Style Sheet Support for Hypermedia Documents". *Proceedings of ACM Hypertext 97.* pp. 216-217, 1997.

Greg J. Badros, Alan Borning, Kim Marriott, and Peter Stuckey. "Constraint cascading style sheets for the web." *In Proceedings of the 1999 ACM Conference on User Interface Software and Technology.* pp.73-82, 1999.

WANG, P. S. "Design and Protocol for Internet Accessible Mathematical Computation." *In Proc. ISSAC'99, ACM Press.* pp. 291-298, 1999.

P. Wadler. "A Formal Semantics of Patterns in XSLT." *In Proceeding of the Conference for Markup Technologies*, 1999.

F. Bry and M. Kraus. "Adaptive Hypermedia made simple using HTML/XML Style Sheet Selectors." *In 2nd Int. Conf. on Adaptive Hyperw, edia and Adaptive Web Based Systems*, 2002.

S. Abiteboul, P. Buneman, and D. Suciu. "*Data on the Web: From Relations to Semistructured Data and XML*." Morgan Kaufmann Publishers, 2000.

Nick Drakos. (2001) "All about LaTeX2HTML." Retrieved January 31, 2002 from http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html. 1996.

Goosens M., and Rahtz S. "From LaTeX to HTML and back." *TUGBOAT*. 1995.

Knuth D.E. "The TeXbook. Computer and Typesetting, Vol. B." Reading MA. 1984.

Helmut Kopka and Patrick W. Daly. "A Guide to LaTeX 3rd Ed." *Addison-Wesley.* 1999.

W3C (2000) "Extensible Markup Language (XML)." Retrieved January 20, 2002 from http://www.w3.org/XML.

W3C.(2000) Cascading Style Sheets, level 2 CSS2 Specification. Retrieved January 20, 2002 from http://www.w3.org/TR/REC-CSS2.

W3C.(2001) "W3C's Math Home Page." Retrieved January 20, 2002 from http://www.w3.org/Math/.

W3C.(2001) "VML - the Vector Markup Language." Retrieved January 20, 2002 from http://www.w3.org/TR/NOTE-VML.

W3C.(2001) "XSL Transformations (XSLT) Version 1.0." Retrieved January 20, 2002 from http://www.w3.org/TR/xslt.