Meta stylesheets for the conversion of Mathematical documents into multiple forms

Bill Naylor, Stephen Watt
Ontario Research Center for Computer Algebra
University of Western Ontario
London Ontario CANADA N6A 5B7
{bill,watt}@orcca.on.ca

August 29, 2001

Abstract

It is sometimes necessary to convert mathematical documents into a number of different forms. This may be achieved by applying XSLT stylesheets (specific to the form required) to the document. We describe how *meta stylesheets* may be used to automatically generate the stylesheets which will be applied to the mathematical documents to generate the multiple forms required. The resources required by the meta stylesheets are extended OpenMath Content Dictionaries. We discuss how the presentational information may be stored in the Content Dictionaries for various different notational styles.

1 Introduction

MathML [7] is an emerging standard from the W3C consortium for the representation of mathematical expressions. Its main design goals are:

1) to provide a means of representing "tex quality" presentation (*presentation MathML*) of mathematics on the internet,

2) to provide a means of representing the semantic meaning of mathematics (*content MathML*), again with the major concern that this should be transferable over the internet.

MathML is an XML application, and as such may readily be converted into other document styles using XSLT [11] stylesheets. It has been designed primarily with

the aim of representing mathematics to the level of *K-12*.
Since mathematical notations consist, in general, of a vocabulary of special symbols and a small number of relative positioning primitives, it is fairly well covered by MathML. Problems arise, however, in the representation of the meaning of general mathematical objects.
One of the limitations with MathML as it stands is that the meaning of an element *resides* in the name of the tag. Therefore, to represent mathematical concepts not pre-defined in MathML it is necessary to refer to some external definitions. A resource which may be able to provide such a *definition server* may exist in the form of OpenMath [9] *Content Dictionaries*.
OpenMath is an XML application whose primary goal is to serve as a communication medium for the semantics of mathematical objects. One of its primary concerns is that it should be extensible, as it is a huge task to represent *all* of known mathematics, and indeed we may need to represent mathematics which is not yet defined! Extensibility is provided via content dictionaries which are XML repositories of mathematical information.
As is described in our paper [3] we may use the OpenMath concept to provide a definition resource which may be used for extending MathML. Padovani et al. [4] discuss a more general framework, of which this is a special case.
In this paper we describe how stylesheets may be automatically generated from extended OpenMath Content Dictionaries using meta stylesheets. Each meta stylesheet will effectively define a new format of result document. It would define the context from which information is taken from the Content Dictionary and the context in which it will be placed into the result document, this context information will be embodied by a style sheet which is specific to the Content Dictionary, we shall call these stylesheets annotation stylesheets.
In this manner, a document written using an *extended MathML* set of symbols may be transformed into one of a number of different formats, for example:

1) presentation MathML, where the notation intended is specified in the content dictionary over which the meta stylesheets are applied.

2) Mixed markup, using `semantic` elements specifying presentation MathML and references to OpenMath Content Dictionaries in `annotation-xml` children.

3) content MathML, using `csymbol` elements and definition URLs which point to symbols in OpenMath Content Dictionaries.

One of the major sources of ambiguity in mathematical notation is the ordering in which the arguments to an operator or function occur. This ordering of arguments should be defined somewhere in the Content Dictionary. We propose the declaration of a `Notation` element within the `Description` element of a symbol. This node will utilize `id` and `xref` attributes to specify the ordering intended, this concept shall be made more precise in section 3.
It may also be necessary to cater for applications which do not have the ability to render or otherwise deal with MathML. In order to cater for these applications we also store TEXdata and a graphical image associated with the notation (e.g.

a URI for a `gif`, `jpeg` or `png` image). In section 9 we shall show how we may automatically create configuration files for a notation selection tool which allows translation of content MathML into various presentation MathML variants, where the notational style may be chosen by a user [2].

2 Dealing with Notational Ambiguity

One of the major problems with mathematical presentation is that there is a many to many relationship between presentations of mathematical objects and semantic meanings. Certain presentations have multiple meanings, for example the symbol M^T could mean "The transpose of the matrix M" or "M to the power T".
Conversely we see that there are many presentations for some mathematical objects, for example the function describing the number of different ways in which m items may be selected from a set of n distinct objects, may be presented as:
n choose m, $\binom{n}{m}$, ${}_nC^m$, C_n^m, C_m^n, binomial(n,m) or Binomial[n,m].
These last two are the notational styles of the computer algebra systems Maple and Mathematica respectively.
This attribute of mathematical presentation (which has its root in the varied history of mathematics) makes extraction of semantic meaning from presentational markup a non-trivial task, due to the ambiguities that it implies. We shall propose a method in which one may specify markup which is both presentational in that it may be converted into presentation MathML (see section 6.1.1) and semantical in that it may be converted into content MathML (see sections 6.1.2 and 6.1.4). This technique utilises a `Notation` element, which we shall detail in the following section.

3 Catalogs of Annotations for Mathematics

In order that we can disambiguate the ordering of arguments to a mathematical operator or function, we have introduced the `Notation` element for symbols defined in an extended OpenMath ContentDictionary. The `Notation` element (in its present form) should satisy the dtd fragment:

```
<!ELEMENT Notation ((version)* | (semantic_template)+)>
```

Whilst the `version` element should satisy the dtd fragment:

```
<!ELEMENT version (image | tex | math)>
```

For our purposes, the different parts of these elements will have the following meanings:

version : There should be one of these elements for every version of the notation (see section 2). It takes the required attribute `precedence`, to specify its order of precedence and the optional attribute `style` which is used as an index for the given style. If no `style` attribute is given (this case should only occur once within any given `Notation` element) then this is the default style. It should have three children: `image`, `tex` and `math`.

semantic_template :
This should have an `OMOBJ` first child, followed by some optional λ function bindings. (constructed within `OMBIND` elements). The element satisfies the dtd fragment:

```
<!ELEMENT semantic_template (OMOBJ | (OMBIND)*)>
```

This element is intended to form a template which may be used to endow the presentation MathML elements (within its `math` siblings) with semantic meaning. This is described in greater detail in section 4.

image : this should have one `src` attribute whose value is the URI of a graphical image file illustrating the notation . This would be useful for display in pulldown menus (and the like) where it is unlikely that MathML-Presentation rendering will soon be available.

tex : this should be a TEX-like representation of the notation. We shall describe this in more detail in section 8. It is included for automatic generation of TEXmarkup from the Extended MathML.

math : this should be a presentation-MathML version of the notation. The elements representing the arguments in this element should have `xref` attributes associated with them, these should have corresponding `id` attributes that are associated with the argument children of the `semantic_template` child of the `Notation` element.

Detailed Example

We now give a detailed example of a `Notation` element which has several `version` children.

Example: 1 We give the example of the `Notation` element for the `choose` symbol. We note in particular the ambiguity that would otherwise be implied by the Russian versus the French styles viz. C_n^m, C_m^n or `style="3"`, `style="4"` respectively

```
<Notation>
  <version precedence="200">
    <image src="choose1.gif"/>
    <tex>#1~{\rm choose}~#2</tex>
```

```
  <math><mrow>
    <mi xref="argChoose1">n</mi>
    <mtext>  </mtext>
    <mtext>choose</mtext>
    <mtext>  </mtext>
    <mi xref="argChoose2">m</mi>
  </mrow></math>
</version>

<version precedence="1000" style="1">
  <image src="choose2.gif"/>
  <tex>\left (\begin{array}{c}#1\\#2\\ \end{array} \right )</tex>
  <math><mrow>
    <mfenced><mtable>
      <mtr><mi xref="argChoose1">n</mi></mtr>
      <mtr><mi xref="argChoose2">m</mi></mtr>
    </mtable></mfenced>
  </mrow></math>
</version>

<version precedence="900" style="2">
  <image src="choose3.gif"/>
  <tex>_#1C^#2</tex>
  <math><mrow>
    <mmultiscripts>
      <mi>C</mi>
      <none/>
      <mi xref="argChoose2">m</mi>
      <mprescripts/><mi xref="argChoose1">n</mi>
      <none/>
    </mmultiscripts>
  </mrow></math>
</version>

<version precedence="900" style="3">
  <image src="choose4.gif"/>
  <tex>C_#2^#1</tex>
  <math><mrow>
    <msubsup>
      <mi>C</mi>
      <mi xref="argChoose2">m</mi><mi xref="argChoose1">n</mi>
    </msubsup>
  </mrow></math>
</version>

<version precedence="900" style="4">
  <image src="choose5.gif"/>
  <tex>C_#1^#2</tex>
  <math><mrow>
    <msubsup>
```

```
        <mi>C</mi>
        <mi xref="argChoose1">n</mi><mi xref="argChoose2">m</mi>
      </msubsup>
    </mrow></math>
  </version>

  <version precedence="800" style="5">
    <image src="choose6.gif"/>
    <tex>{\rm binomial}(#1,#2)</tex>
    <math>
      <mi>binomial</mi><mo>&ApplyFunction;</mo>
      <mfenced>
        <mi xref="argChoose1">n</mi><mi xref="argChoose2">m</mi>
      </mfenced>
    </math>
  </version>

  <version precedence="800" style="6">
    <image src="choose7.gif"/>
    <tex>{\rm Binomial}[#1,#2]</tex>
    <math>
      <mi>binomial</mi><mo>&ApplyFunction;</mo>
      <mfenced open="[" close="]">
        <mi xref="argChoose1">n</mi><mi xref="argChoose2">m</mi>
      </mfenced>
    </math>
  </version>

  <semantic_template><OMOBJ><OMA>
    <OMS cd="combinat4" name="choose"/>
    <OMV name="n" id="argChoose1"/>
    <OMV name="m" id="argChoose2"/>
  </OMA></OMOBJ></semantic_template>
</Notation>
```

We may specify the presentation of a particular version of a symbol by giving a specific value to the `style` attribute of the extended MathML symbol, where the attributes value is the same as the value of the `style` attribute associated with the particular `version` element required.

4 Semantic Templates for Mathematical Objects

In order to specify notation for Mathematical objects it is necessary to link the presentation information with some formal specification of the object. This is done through the XPath [10] `id`, `xref` linking mechanism. In most cases this is quite a simple procedure and is covered in subsection 4.1. However in some cases this process is not sufficient to specify all of the objects refered to in the notation, in

this case extra technology is required, we cover this in subsection 4.2.
Whithin the presentation information which is represented by presentation MathML, there are elements representing the differrent parts of the object being represented. We attach `id` attributes to each of these elements.

Example: 2 *In this example the* `m` *subscript of* `C` *is labeled with the attribute* `xref` *which has a value of* `argChoose2` *whilst the superscript* `n` *has an* `xref` *attribute with a value of* `argChoose1`.

```
<math><mrow>
  <msubsup>
    <mi> C </mi>
    <mi xref="argChoose2"> m </mi><mi xref="argChoose1"> n </mi>
  </msubsup>
</mrow></math>
```

Now it is necessary to provide a *semantic template* that the parts of the presentational information can be linked to. The first step is to provide a *prototype object* which has *anchors* which may be linked to by the presentation.

4.1 Function Application

The most common type of mathematical object is simply that of a function being applied to a number of arguments. In this case the prototype is just the OpenMath Application (`OMA`) of the function to OpenMath Variables (`OMV`s) in the place of its arguments. An `id` attribute is attached to each of the arguments. These attributes should have the same values as the `xref` attributes in the corresponding parts of the presentation information.

Example: 3 *We give an example of a template which could be used with the above example 2.*

```
<semantic_template><OMOBJ><OMA>
  <OMS cd="combinat4" name="choose"/>
  <OMV name="n" id="argChoose1"/>
  <OMV name="m" id="argChoose2"/>
</OMA></OMOBJ></semantic_template>
```

4.2 Semantic Templates which include extra information

There are however cases where subelements of the presentation do not correspond to specific elements in any generic formulation of the formal object (typical examples are n'ary functions, matrices and partial differentiations). We propose a scheme

whereby we may prescribe complete (in that they include all the information required to construct the given notations), unambiguous templates that include anchors for the presentation sub-elements. The scheme we use is to follow the prototype object with functions which calculate the required values (In the case of n'ary functions this would be the elements of the n'tuple, in the case of a matrix, the elements of the matrix, partial differentiation is more complex and shall be investigated in Appendix A), we shall call these functions *template functions*. The idea is that the `OMBIND` element enclosing the function has an `id` attribute which may be linked to from the presentation. The expression part of the function definition will link to the prototype in the usual way using `xref` attributes.

4.3 Extracting the template function call from the presentation

It is necessary to include the arguments to the template functions in the document, where their positions relative to the element which specifies the function call (via an `xref` attribute) is constant. This shall be done following the same patern as the `OMA` (OpenMath function Application) element. The first child is arbitrary (this is included so that valid Presentation MathML may be used), the following children are the arguments.

Example: 4 *We give an example of a matrix, which we wish to be displayed as:*

$$\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

In order to provide anchors for the markup which enables this, as part of the `semantic_template` *we require a function taking two variables i, j. The return value of this function should be the element indexed by i and j. In this case, the top left, top right, lower left and lower right elements. Presentation MathML which could be used for this first form follows:*

```
<mrow>
  <mo> ( </mo>
  <mtable>
    <mtr>
      <msub xref="matelt">
        <mi>x</mi> <mrow><mn>1</mn> <mn>1</mn></mrow>
      </msub>
      <mi>&hellip;</mi>
      <msub xref="matelt">
        <mi>x</mi> <mrow><mn>1</mn> <mi>n</mi></mrow>
      </msub>
    </mtr>
    <mtr>
```

```
      <mi>&vellip;</mi><mi>&#x022F1;</mi><mi>&vellip;</mi>
    </mtr>
    <mtr>
      <msub xref="matelt">
        <mi>x</mi> <mrow><mn>m</mn> <mn>1</mn></mrow>
      </msub>
      <mi>&hellip;</mi>
      <msub xref="matelt">
        <mi>x</mi> <mrow><mn>m</mn> <mi>n</mi></mrow>
      </msub>
    </mtr>
  </mtable>
  <mo> ) </mo>
</mrow>
```

(The entity references `…`*,* `⋮` *and* `⋱` *are the unicode references for the characters* $\cdots$*,* $\vdots$ *and* $\ddots$ *respecitvely.*

Now the function in the `semantic_template` *in mathematical notation is the following:*

$$\lambda : [i, j] \mapsto M.i.j$$

where λ *has an* `id` *attribute with value* `matelt` *and* M *is linked to the prototype. The full* `notation` *element is given in Appendix B*

4.4 Multiple Templates

There are certain situations where the expressibility of an OpenMath symbol is so broad that the same OpenMath symbol may be used to represent structurally disparate concepts. In order to deal with these incongruous symbols, it is necessary to provide more than one `semantic_template` element for the `Notation` element.

Example: 5 *The example we shall look at is that of definite integration. This concept is represented by the OpenMath symbol* `defint` *from the ContentDictionary* `calculus1`*. The symbol may be applied over a "range (e.g. a set) of integration". This leaves the freedom that the symbol may be applied over an interval (with upper and lower bounds a,b respectively) in which case we might intend that it should be displayed as* $\int_b^a f(x)dx$*, or it may be applied over a set (denoted by S) in which case we may intend that it should be displayed as* $\int_{x \in S} f(x)dx$*.* `semantic_template`*s for these cases follow.*
In the case of $\int_b^a f(x)dx$*:*

```
<semantic_template><OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="defint"/>
    <OMA>
      <OMS cd="interval1" name="interval"/>
```

```
      <OMV name="b" id="intervalDefintArg1"/>
      <OMV name="a" id="intervalDefintArg1"/>
    </OMA>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="x"/>
      </OMBVAR>
      <OMA>
        <OMV name="f"/>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ></semantic_template>
```

In the case of $\int_{x \in S} f(x)dx$*:*

```
<semantic_template><OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="defint"/>
    <OMV name="S" id="setDefintArg"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="x"/>
      </OMBVAR>
      <OMA>
        <OMV name="f"/>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ></semantic_template>
```

these different `semantic_template` *elements will both be children of the* `Notation` *element for the* `defint` *symbol.*

5 Default Notations

We wish the user to have full choice over which style is used within their document, whilst retaining the semantic content. This may be achieved using the `version` with `style` attributes and `semantic_template` elements described earlier. However we would like the user to have the option to relegate the styling decisions to some defaulting mechanisms. We specify a heirarchy of defaults which enable this:

- **priority 1**: Command line control,

  This is the level of control over notational defaults with the highest precedence. When the user applies the generated style sheets to the input document, written in Extended MathML, default styles may be specified on the command line. These will override any defaults given at other levels of the default heirarchy. possible uses for this default level include:

  1) production of copies of documents to be sent to different countries which have different or even conflicting styles in common use.
  2) production of copies of documents to be submitted to separate journals which require differing styles.

- **priority 2**: Input Document defaults,

  This level of control over notational defaults has an intermediate precedence level. This allows an author to personalise a document with his own notational styles. An element must be included at the top level of the input document, which has the following syntax:

```
<default_style>
  <for symbol="symbName" style="rqdStyle"/>
   ...
</default_style>
```

  here the element `default_style` satisfies the dtd fragment:

```
<!ELEMENT default_style (for)*>
```

  and the element `for` satisfies the dtd fragment:

```
<!ELEMENT for EMPTY>
<!ATTLIST for symbol CDATA #REQUIRED
              style CDATA #REQUIRED>
```

- **priority 3**: Meta Style Sheet defaults,

  This level of control over notational defaults has a lower intermediate precedence level. It is unlikely to be used frequently. It might be used if a particular meta style sheet distribution was going out to a particular market which shared a common notational language. e.g. A meta style sheet going out to an engineering market might specify that the symbol for the square root of -1 was j, whereas for mathematicians it might be i.

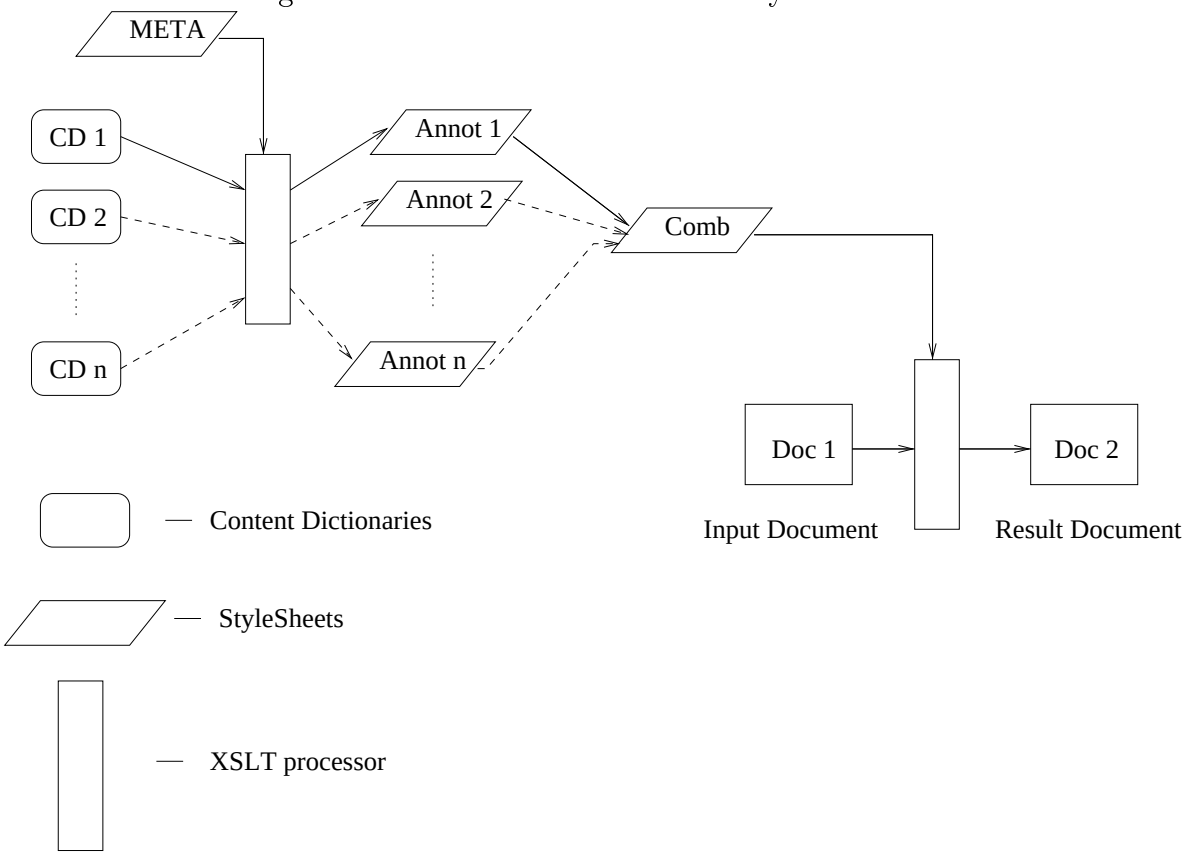
- **priority 4**: Content Dictionary defaults,

  This default level is the lowest priority but the level which must be adhered to if no other default is given and if no style has been set in the input document. These defaults are required to be given.

6 Automatic Creation of Stylesheets

In this section we describe how given a set of OpenMath content dictionaries, we may automatically create a set of stylesheets that may be used to transform mathematical documents. Our method is described diagrammatically in figure 1.

Figure 1: Automatic Production of Stylesheets



- The Content Dictionaries are denoted *CD 1* to *CD n*.
- The Meta style sheet (a different one for each format required) is denoted *META*.
- The annotated style sheet that is the result of applying *META* to *CD i* is denoted *Annot i*.
- All the annotated style sheets are imported by a combining style sheet denoted *Comb*.
- Finally a document *Doc 1* which is written in the extended MathML implied by the content dictionaries *CD 1* to *CD n*, may be translated by the stylesheet *Comb* into a document *Doc 2* which is in the required format.

6.1 The different formats we have considered

We now make a more detailed presentation of the different formats we have considered by considering the translation (of the application of the `choose` symbol to the variables n and m) to each of these formats. The different formats we shall consider are:

- Extended Presentation MathML → Presentation MathML.
- Extended Content MathML → Content MathML with OpenMath references.
- Extended Presentation MathML → semantics elements with Presentation MathML and OpenMath annotations.
- Extended Content MathML → csymbol elements with Openmath symbol URLs.

6.1.1 Extended Presentation MathML → Presentation MathML

In order to perform this transformation the annotated stylesheet must include the following constructs in the templates for each symbol:
We shall assume that the symbol which matches the template in question is `xmml:choose`[1].

```
<xsl2:choose>
  <xsl2:when test="@style=style1">
    -- presentation information taken from the 'style1' style
       of xmml:choose --
  </xsl2:when>

  ... one when element for each non-default style ...

  <xsl2:otherwise>
    -- presentation information taken from the default style
       of xmml:choose --
  </xsl2:otherwise>
</xsl2:choose>
```

Each of the 'presentation information' sections will include `apply-template` elements to specify the presentation of the arguments to the symbol, however these will not necessarily be in the order in which they appear in the presentation. They will be in the order in which the arguments are given in the `semantic_template` child of the `Notation` element (linked by the `xref`, `id` attributes).
These stylesheets must be generated by the meta style sheets. We see that all of the information necessary to create the choose elements exists in the content dictionaries.

[1] we use the xmml prefix here in order to emphasis the differences between the extended MathML symbol (`xmml:choose`) and the XSLT symbol (`xsl2:choose`)

If a symbol exists in an OpenMath Content Dictionary with the name of `Symb`, the corresponding extended MathML element will have the same name. This element should be the first child of an `mrow` element. The following children should be the arguments of the function.

Example: 6 If we wish to specify, within some extended markup, the presentation of the *choose* function applied to two arguments n and m, we should use the following markup:

```
...
<mrow>
  <choose/> <!-- no style attribute, so the default is assumed -->
  <mi>n</mi> <!-- these could be arbitrary arguments -->
  <mi>m</mi>
</mrow>
...
```

N.B. the default referred to in the above depends on the default level which has been selected from the default heirarchydescribed in section 5. On application of the style sheet which has been automatically created from the content dictionary which contains the choose symbol, we obtain the markup:

```
<mrow>
  <mi xref="argChoose1">n</mi>
  <mtext>  </mtext>
  <mtext>choose</mtext>
  <mtext>  </mtext>
  <mi xref="argChoose2">m</mi>
</mrow>
```

The rendering will be approximately "n choose m".

The reader is referred back to section 5 for a discussion on how the default style may be altered.

Example: 7 We now give an example where one particular style is required. If we required markup to display the same application as in example 6 but in style "2", for example, we could use the markup:

```
<mapply>
  <choose style="2"/>
  <mi>n</mi>
  <mi>m</mi>
</mapply>
```

This would be converted by the same style sheet as used in example 6 to the following:

```
<mrow>
  <mmultiscripts>
    <mi>C</mi>
    <none/>
    <mi xref="argChoose2">m</mi>
    <mprescripts/><mi xref="argChoose1">n</mi>
    <none/>
  </mmultiscripts>
</mrow>
```

The rendering will be approximately "${}_nC^m$".

6.1.2 Extended Content MathML → Content MathML with Openmath references

In order to perform this transformation, not much extra information is required, the required form is the following (this is based on the `semantics` element, a mapping from some markup to an external definition resource):

```
<apply>
  <semantics>
    <!--extended content MathML symbol-->
    <annotation-xml>
      <!--OpenMath reference-->
    </annotation-xml>
  </semantics>
  <!--arguments-->
</apply>
```

The extended content MathML symbol will be the same as in the input document, so that can just be copied through. This can also be done for the arguments. The only section which needs any extra information is the OpenMath reference. This will be an `<OMS cd="?" name="?"/>` the `name` attribute value may be determined trivially from the extended content MathML symbol name, and the `cd` attribute value may be read off from the content dictionary, as it must be given as one of the first level children elements. The only remaining task is to wrap the determined elements in `semantics`, `annotation-xml` and `apply` elements where appropriate.

If a symbol exists in an OpenMath Content Dictionary with the name of `Symb`, the corresponding extended MathML element will be `Symb`. This element should be the first child of an `apply` element. The following children should be the arguments of the function.

Example: 8 If we wish to specify within some extended content markup the application of the choose function to two arguments n and m, we should use the following markup:

```
...
<apply>
  <choose/>
  <ci>n</ci>
  <ci>m</ci>
</apply>
...
```

The resulting markup that we would obtain after application of the style sheet that has been automatically created from the content dictionary which contains the `choose` symbol (which for arguments sake was the content dictionary `series.ocd`) is:

```
...
<apply>
  <semantics>
    <choose/>       <!-- extended content MathML symbol -->
    <annotation-xml encoding="OpenMath">
      <om:OMS cd="series" name="choose"/>     <!-- OpenMath reference -->
    </annotation-xml>
  </semantics>
  <ci>n</ci>        <!-- arguments -->
  <ci>m</ci>
</apply>
...
```

6.1.3 Extended Presentation MathML → semantics elements with Presentation MathML and OpenMath

In order to perform this transformation essentially we only need to perform a combination of the transformations of sections 6.1.1 and 6.1.2, to obtain a document with the structure:

```
  <semantics>
    <!-- MathML Presentation of the application -->
    <annotation-xml>
      <!-- OpenMath application -->
    </annotation-xml>
  </semantics>
```

6.1.4 Extended Content MathML → csymbol elements and Openmath symbol URLs

The only information that is required for this transformation is a URI of the symbol, this can be determined from one of the top level children in the content dictionaries, the `CDURL` element and the name of the symbol, (which problem we solved in section 6.1.2). This URI must be given as the value of the `csymbol` element.
The syntax with which we wish to specify the extended content MathML shall be the same as in example 8.

Example: 9 We shall also use the same input as in example 8. However we should expect to get the output:

```
<apply>
  <csymbol definitionURL=
    "http://orcca.on.ca/~bill/OpenMath/SpecialFns/html/cd/series.html#choose"/>
  <ci>n</ci>
  <ci>m</ci>
</apply>
```

7 Stylesheets generated from Template Functions

Clearly it is an impossible task to write a meta style sheet which constructs a style sheet to calculate the evaluation of a generic function. However certain values are returned from template functions with such frequency that it may be a good idea to implement meta style sheets which implement the calculation of these particular objects. The most common of these is:

- the selection of the *i'th* element from a list or collection of objects.
- the selection of the *i,j'th* element of an array of values ar a matrix.
- some object which specifies the repetition of a sequence of commands over a collection of object. It seems that it would not be possible to specify how to display every part of an agregate object without this sort of return value.

These issues reflect current research.

8 Specification of the TEXlike representation

We now look at the specification of the string which is to be held in the `tex` elements. These elements may contain *Character Data*. This character data will be expected

to be TEXsource, except for the parts which are designated as representing the arguments of the function. For each argument we must store three fields, these are the following:

field 1: This is equivalent to the `xref` attribute in the `math` element. It should hold the same value.

field 2: This holds an arbitrary value which may be used to show the object in a generic setting.

field 3: This holds zero or more arguments which will be interprested as the arguments for template functions, if field 1 is pointing to a template function.

This is done by specifying the TEXtemplate in the following manner:
<tex>
⋆⋆⋆/arg{val1}{default1}[arg$_1$1,$\cdots$,arg$_1$m$_1$]
⋆⋆⋆/arg{valn}{defaultn}[arg$_n$1,$\cdots$, arg$_n$m$_n$]⋆⋆⋆
</tex>
In the above: ⋆⋆⋆ represents arbitrary TEXsource, val$i$ is the value of the $i'th$ xref attribute, default$i$ is the $i'th$ arbitrary value and the values arg$_i$j are arguments to template functions (if the link is not to a template function then the mthird field will be empty.

9 Applications

In the next section we shall consider a tool which has been written and for which our method finds useful application.

9.1 A Notation selection tool

The tool we consider is called "A Notation Selection Tool For MathML" [2]. This tool is "an extensible notation selection tool, which provides a graphical user interface (GUI) to allow the user to select his/her notational conventions while transforming content MathML to presentation MathML". The tool is extensible in that it references an XML file to extract graphical information, and XSLT templates in performing its transformation. Our method may be used to create these files (modulus the relevant `Notation` elements being available).

9.2 Structure of the input file for the Notation Selection Tool

The XML document that is required by the Notation Selection Tool (we shall denote these *Notation input document*s) must have the following structure (In this example

we assume that the relevant content dictionary is called `Arithmetic` containing symbols; for example `DIVISION`).

```
<?xml version="1.0"?>
<mnotations>
  <catalog>  <!-- Arithmetic -->
    <name> Arithmetic </name>
    <itemlist>
      <item>
        <keyword> DIVISION </keyword>
        <choicelist>
          <choice>
            <image src="div1.gif"/>
            <presentation>
              <!-- XSLT templates for this notation -->
            </presentation>
          </choice>
            .
            .  <!-- other choices for DIVISION -->
            .
        </choicelist>
      </item>
        .
        .  <!-- other items of Arithmetic -->
        .
    </itemlist>
  </catalog>
    .
    .  <!-- other catalogs -->
    .
</mnotations>
```

We may construct a `META` style sheet which produces one Notation input file for each content dictionary. The correspondance we make between the elements of the content dictionary and the elements of the Notation input document are the following:

Content Dictionary	Notation Input Document
CD	catalog
CDName	name
CDDefinition	item
Name	keyword
Notation	choicelist
version	choice
image	image
math/semantic_template	presentation

It must be noted that the information which resides in the `presentation` element of

the Notation Input Document will originate from three sources, the `math` elements of the content dictionaries, the `semantic_template` elements of the content dictionaries and the `META` stylesheets. The one element of the Notation Input Document that we have not dealt with here is the `keyvalue` element. The function of this element is to identify the particular style of notation required, this information is held in the `style` attribute of the `version` element using our scheme. We show a screen shot of the tool in action in figure 2.

10 Further directions

The problems which become apparent when we try to specify notations with an arbitrary number of elements, e.g. an arbitrary length vector or matrix, have not been addressed. This is an area we are working on.
It would be possible to write a tool which could translate between Extended MathML and LaTeX, by utilizing the information held in the `tex` element of the notation element.

11 Conclusion

If a comprehensive set of content dictionaries written in this extended OpenMath where constructed, using these ideas it would be possible to write presentational markup which had real semantic meaning and could be automatically converted into presentational MathML or some form of content MathML.

12 Addendum

These ideas are at a very early stage of development and it is certain that collaboration with certain standards and bodies (most notably, the OpenMath consortium [9], the MathML standard [7] and the OMDoc project [8]) will occur, in particular with regard to finalizing the names of various elements and attributes.

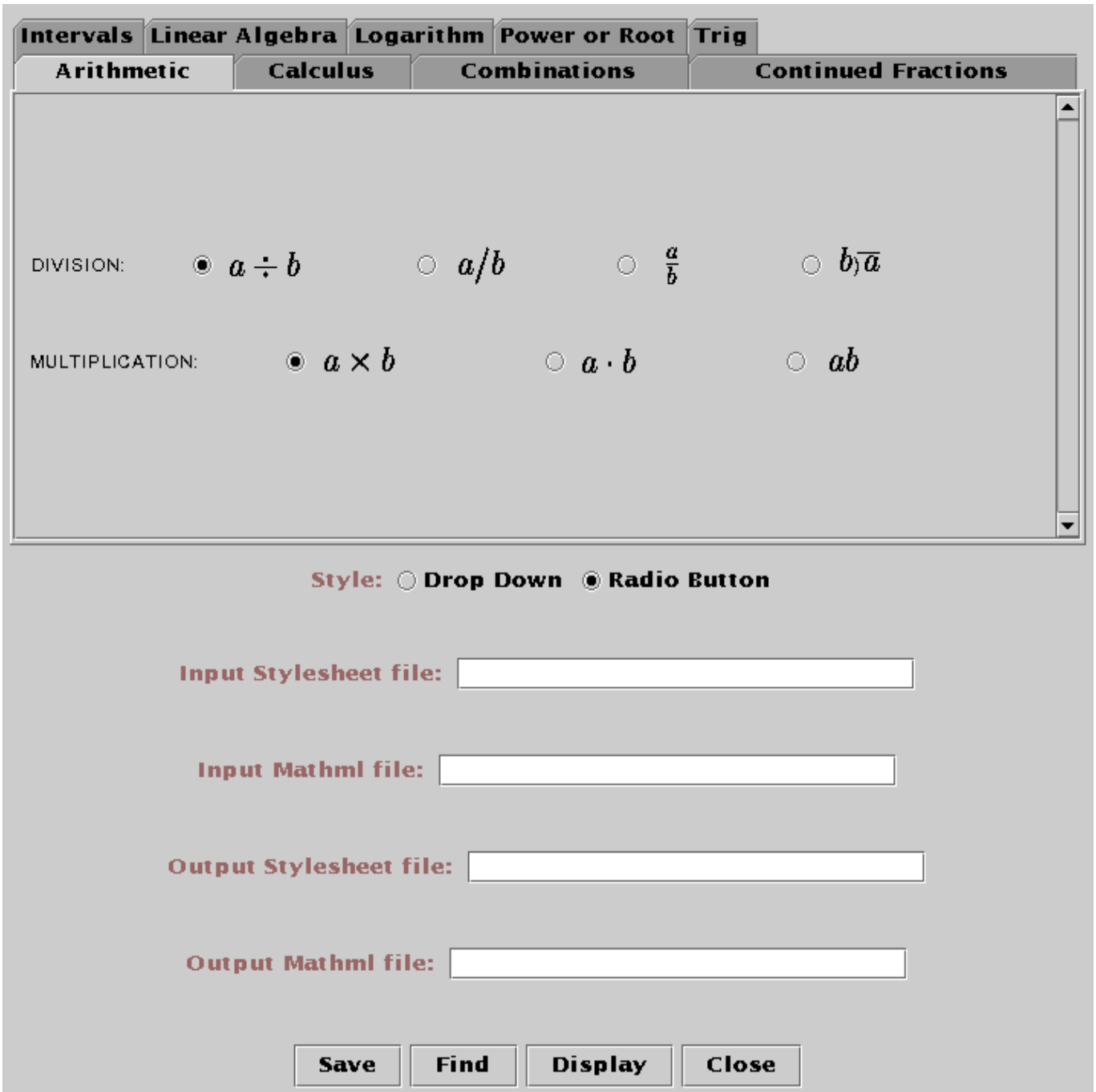


Figure 2: A screen shot showing possible choices for division and multiplication

A Partial Differentiation example

[2]We display a notation element for the partialdiff symbol:

```
<Notation>
  <version>
  <math>
    <mrow>
      <mfrac>
        <msup><mi>&part;</mi><mi xref="sum">n</mi></msup>
        <mrow>
          <msubsup><mi>&part;</mi>
            <msub xref="getVar">
              <mi>x</mi><mn>1</mn>
            </msub>
            <msub xref="getDeg"><mi>n</mi><mn>1</mn></msub>
          </msubsup>
          <mi>&#x02026;</mi>
          <msubsup><mi>&part;</mi>
            <msub xref="getVar">
              <mi>x</mi><mi>m</mi>
            </msub>
            <msub xref="getDeg"><mi>m</mi><mn>1</mn></msub>
          </msubsup>
        </mrow>
      </mfrac>
      <mi>f</mi>
      <mo>&ApplyFunction;</mo>
      <mfenced>
        <msub xref="getVar">
          <mi>x</mi><mn>1</mn>
        </msub>
        <mi>&#x02026;</mi>
        <msub xref="getVar">
          <mi>x</mi><mi>m</mi>
        </msub>
      </mfenced>
    </mrow>
  </math>
  </version>
  <semantic_template>
    <OMOBJ> <!-- this OMOBJ is the prototype -->
      <OMA>
        <OMS cd="calculus1" name="partialdiff2"/>
        <OMV name="listOfLists" id="ll"/>
<!-- this list of lists is a list of pairs where the first element of
      each pair is an index to the variable and the second is the
```

[2]We note here that the symbol `list_selector` from the content dictionary `list2` is not part of the OpenMath standard at present and is only proposed

```
      order of the differentiation in that variable -->
        <OMV name="f" id="functionName"/> <!-- this is the function -->
      </OMA>
    </OMOBJ>
<!-- the following function allows selection of the variables -->
    <OMBIND id="getVar">
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="n"/>
      </OMBVAR>
      <OMA><OMS cd="list2" name="list_selector"/>
        <OMI> 1 </OMI>
        <OMA><OMS cd="list2" name="list_selector"/>
          <OMV name="n"/>
          <OMV name="ll" xref="ll"/>
      </OMA></OMA>
    </OMBIND>
<!-- the following function allows selection of the degrees -->
    <OMBIND id="getDeg">
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="n"/>
      </OMBVAR>
      <OMA><OMS cd="list2" name="list_selector"/>
        <OMI> 2 </OMI>
        <OMA><OMS cd="list2" name="list_selector"/>
          <OMV name="n"/>
          <OMV name="ll" xref="ll"/>
      </OMA></OMA>
    </OMBIND>
<!-- the following function calculates the sum of selected items from
  a list of tuples (in this case it will be the order of a function) -->
    <OMBIND id="order">
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR></OMBVAR> <!-- the function is nullary -->
      <OMA>
        <OMS cd="arith1" name="sum"/>
        <OMA>
          <OMS cd="interval1" name="integer_interval"/>
          <OMI> 1 </OMI>
          <OMA>
            <OMS cd="list2" name="size"/>
            <OMV name="listOfLists" xref="ll"/>
          </OMA>
          <OMBIND>
            <OMS cd="fns1" name="lambda"/>
            <OMBVAR>
              <OMV name="i"/>
            </OMBVAR>
            <OMA>
```

```
              <OMS cd="list2" name="list_selector"/>
              <OMI> 2 </OMI>
              <OMA>
                <OMS cd="list2" name="list_selector"/>
                <OMV name="i"/>
                <OMV name="listOfLists" xref="ll"/>
              </OMA>
            </OMA>
          </OMBIND>
        </OMA>
      </OMA>
    </OMBIND>
  </semantic_template>
</Notation>
```

B Matrix example

We display a possible notation element for the matrix symbol:

```
<Notation>
  <version precedence="100">
    <image src="matrixNotationDefault.gif"/>
    <tex>
\left (\begin{array}{ccc}
\arg{matelt}{x}[1,1]&\cdots&\arg{matelt}{x}[1,n]\\
\vdots&\ddots&\vdots\\
\arg{matelt}{x}[m,1]&\cdots&\arg{matelt}{x}[m,n]
\end{array}\right )
    </tex>
    <math>
    <mrow>
      <mo> ( </mo>
      <mtable>
        <mtr>
          <msub xref="matelt">
            <mi>x</mi> <mrow><mn>1</mn> <mn>1</mn></mrow>
          </msub>
          <mi>&hellip;</mi>
          <msub xref="matelt">
            <mi>x</mi> <mrow><mn>1</mn> <mi>n</mi></mrow>
          </msub>
        </mtr>
        <mtr>
          <mi>&vellip;</mi><mi>&#x022F1;</mi><mi>&vellip;</mi>
        </mtr>
        <mtr>
```

```
          <msub xref="matelt">
            <mi>x</mi> <mrow><mn>m</mn> <mn>1</mn></mrow>
          </msub>
          <mi>&hellip;</mi>
          <msub xref="matelt">
            <mi>x</mi> <mrow><mn>m</mn> <mi>n</mi></mrow>
          </msub>
        </mtr>
      </mtable>
      <mo> ) </mo>
    </mrow>
    </math>
  </version>
  <semantic_template><OMOBJ>
    <OMV name="M" id="matrix"/>  <!-- this variable represents the matrix -->
    <OMBIND id="matelt">
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="i"/>
        <OMV name="j"/>
      </OMBVAR>
      <OMA>
        <OMS cd="linalg1" name="matrix_selector"
        <OMV name="i"/>
        <OMV name="j"/>
        <OMV name="M" xref="matrix"/>
      </OMA>
    </OMBIND>
  </OMOBJ></semantic_template>
</Notation>
```

References

[1] M.Abramotiz and A.Stegun: Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables.

[2] Dicheng Liu: A notation selection tool for MathML stylesheets, University of Western Ontario MSc Project, 2001.

[3] Bill Naylor, Stephen Watt: On the relationship between OpenMath and MathML, 2001 Workshop on Internet Accessible Mathematical Computation: http://icm.mcs.kent.edu/research/iamc2001.papers/nay.ps.gz

[4] Luca Padovani, Irene Schena and Stephen Watt: Stylesheets for Mathematics in the Semantic Web (preprint).

[5] Igor Rodionov: Tools for MathML, University of Western Ontario MSc Thesis, 2001.

[6] Amaya: http://www.w3.org/Amaya

[7] MathML: http://www.w3.org/TR/MathML2/

[8] OMDoc: http://www.mathweb.org/omdoc/

[9] OpenMath: http://www.openmath.org

[10] XPath: http://www.w3.org/TR/xpath

[11] XSLT: http://www.w3.org/TR/xslt.html