Notation Selection in Mathematical Computing Environments

Elena Smirnova Stephen M. Watt

Abstract

We examine the problem of notation selection in mathematical computing environments. Users of mathematical software may require different notations for the same expression in a variety of settings. How this can be managed in a general way is the subject of this paper. We describe a software tool that can be configured to allow mathematical packages to provide output according to specified notation preferences. We explore how the choice of a set of notations can be used to disambiguate mathematical input and output in a variety of settings, including mathematical handwriting recognition, mathematical knowledge management and computer algebra systems.

1 Introduction

One of the problems for natural use of mathematical expressions with a computer is the absence of unique relationship between the syntactic presentation of mathematical objects and their semantic meaning. On one hand, the *same notation* can denote several *different mathematical objects*. For example, $\tan^{-1}$ in $\tan^2(x) + \tan^{-1}(x)$ may mean "cotangent" or "arctangent". On the other hand the *same mathematical object* can be written using *several different notations*: for instance, inner product has several notations, including: $\langle p, q \rangle$, $p \cdot q$ and $p \mathbin{\mathsf{L}} q$.

Notational ambiguity leads to possible misinterpretation of mathematical content not only by human reader, but also by automated tools, such as computer algebra systems, theorem provers, mathematical data format converters and pen-based applications. Misunderstanding of a mathematical notation will lead to unexpected or incorrect results, which in the case of automated environments can be hard to track down. All of this suggests the utility of additional tools to allow selection of default and preferred notations for use within mathematical software environments and in human-computer interfaces. This paper presents our approach to this problem.

Previous work on notation selection in mathematical context were reported in [5], [6], [11] and [17]. In this paper we elaborate on an approach to mathematical notation selection based on the idea of using meta-stylesheets for the conversion of mathematical documents [14]. We present an implementation in the form of a Notation Selection Tool for the XML-based mathematical data formats MathML[2] and OpenMath[3].

The rest of the paper is organized as follows: Section 2 gives some examples of common notational ambiguities and their origins. Section 3 describes our implementation of a Notation Selection Tool. In Section 4 we go on to consider domains of application for the notation selection technique and suggest how mathematical notation can be used for mathematical context management within various environments. The final section presents our conclusions.

2 Mathematical Notation

2.1 Types of ambiguity and their origin

When we deal with written mathematics in various domains, there are often situations where *one notation* is used to represent completely *different mathematical ideas*. For example the expression u' can mean "derivative", "minute", "logical not", "group inverse", "transformation of u", or have any one of several other interpretations. Usually the meaning can be determined from the context, but when several domains of mathematical science are used together, the meaning can be unclear and alternative notations may be required.

Equally well, *one mathematical idea* often will have *several different notations*. For example, the partial derivatives of a function may be presented in any of several ways, including: f_x, f'_x, $\partial_x f$, $\bigtriangledown_x f$, $\frac{\partial f}{\partial x}$ or $\mathrm{D}_x f$. Furthermore, it is not only the different symbols and structures that may cause ambiguity: in many cases the spatial layout of an expression is vital. For example, in various settings $C(n,k)$, ${}_nC^k$, ${}^nC_k$, ${}_nC_k$, C^k_n and C^n_k will be used for the same value – the binomial coefficient "n choose k", $\binom{n}{k}$. While the first four notations can be easily understood, the pair C^k_n and C^n_k present a serious ambiguity.

Multiple notations for the same mathematical concept may exist for a number of reasons:

- The *mathematical context* often affects the usual appearance of a formula, e.g. the fractional power of an expression can be denoted as $\sqrt[n]{f}$ or as $f^{1/n}$.
- The *area of application* may imply a default notation, for example i for $\sqrt{-1}$ in complex analysis vs. j for the same value in electrical engineering. Likewise mathematicians commonly write integration as $\int f(t)\mathrm{d}t$, while in physics the notation $\int \mathrm{d}t f(t)$ is often preferred.
- Sometimes *national and cultural conventions* apply. For example, the tangent function is presented by "tan" in Western Europe and North America but by "tg" in Eastern Europe and China. The open interval, usually denoted as (a, b) in North America, would be written as $]a, b[$ in France and some other countries.
- The *historical period* also leads to different notations. For instance, the older use of lines for grouping, *e.g.* $3\overline{a+b}$, versus the modern use of parentheses, *e.g.* $3(a+b)$.
- The *level of mathematical sophistication* may influence the preferred representation of expressions. For example, $a \div b$ and $b\overline{)\,a}$, are usually used in elementary school arithmetic, while a/b and $\frac{a}{b}$ are used at higher levels.

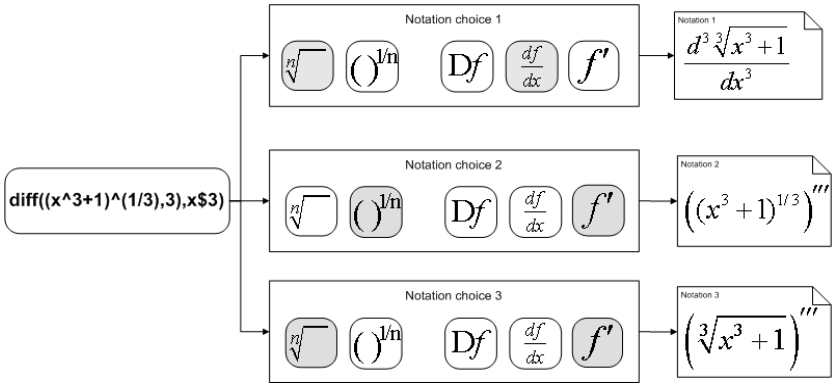

Figure 1: Rendering of mathematical objects in different notations

2.2 Notation decisions

To render a document that contains the *semantics* of mathematical objects, software tools typically use some specific set of given notations. This is the approach typically used in stylesheets for displaying conceptually-oriented Content MathML and OpenMath. Default notions are also used by mathematical software in a "pretty-printing" of computation results and in exporting output to presentational formats, such as LaTeX or Presentation MathML.

While this approach is often satisfactory, users from different scientific and cultural backgrounds may find the choice notation inconvenient. For example, for rendering an expression for "the third derivative of the cube root of the expression x cubed plus one," several notations for each operation can be used (see Figure 1). Depending on the choices for these notations and their combination, the displayed formula may either look crisp and clear or appear to be crowded and unaesthetic. In the worst case, a human reader or software package that is not familiar with a notation may not be able to determine the meaning of an expression. Unless carefully designed and maintained, software that uses fixed default notations can easily use the same notation to render different mathematical concepts. This introduces ambiguities for any later use of the output by human readers or other mathematical software.

Software with custom rendering rules can be used to avoid these ambiguities. Let us consider the ambiguous use of parentheses in the notations $\binom{m}{n}$, (a, b) and $\left(\frac{n}{p}\right)$. The first could be interpreted as a binomial coefficient or column vector. The second could be an ordered pair, an inner product, a row vector, a greatest common divisor or an interval. The third could be a Legendre symbol or a parenthesized fraction. In an output where multiple uses of the same notation would occur, these could be disambiguated by specifying the use of alternatives, *e.g.* C_m^n or $\gcd(a, b)$.

The other common situation is a mathematical concept with many notations. In this case, users may wish to choose certain specific notation for input and output of mathematical expressions. In most cases, however, only one pre-defined syntax will be accepted

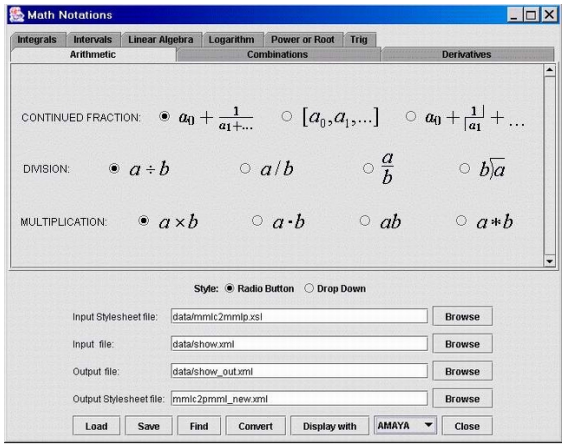

Figure 2: Stand-alone Notation Selection Tool application

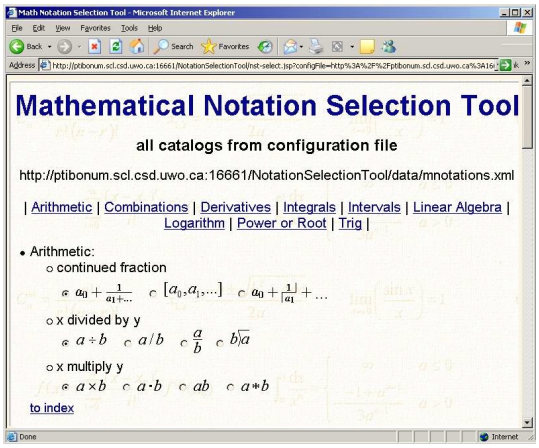

Figure 3: On-line Notation Selection Tool interface

by mathematical software packages. For instance, the notation acos(x) used by Fortran to denote "inverse of the cosine function" will not be recognized by Maple. Having the possibility to advise a mathematical software system on a set of preferred notations can serve both to disambiguate input and to produce more useful output. The collection of selected notations defines a target space of mathematical concepts. We conclude that introducing a technique to select notation for use in mathematical software would help to overcome some difficulties we have described.

3 A Notation Selection Tool

In this section we describe a software tool that allows applications to be configured to employ user-selected notations. Our *Notation Selection Tool* is designed to drive the conversion from XML-based conceptually-oriented mathematical documents into notationally-oriented formats. The Notation Selection Tool does this by generating rules to translate between mathematical expressions in different XML formats. One feature of the Notation Selection Tool is in its extensibility: an application may define the scope of mathematical concepts that tool can handle, as well as the set of alternative notations allowed for each concept.

Our implementation presents a graphical user interface that is used to generate an XSLT stylesheet. This stylesheet may then by used to convert from Content MathML to Presentation MathML. The interface allows the user to select notational conventions for a variety of concepts, organized by mathematical area. The software is deployed in two configurations: a stand-alone Java application (Figure 2) and an on-line service at `http://ptibonum.scl.csd.uwo.ca:16661/NotationSelectionTool` (Figure 3). The core of the Notation Selection Tool is written in Java. The stand-alone version uses the Swing library[9] and the on-line version is implemented with Java Server Pages [8]. The package includes a base XSLT stylesheet for Content MathML to Presentation MathML transformation [7][16], a library of images to present possible notations, and a configuration file.

3.1 Target mathematical domains

To test the feasibility of our approach, we have considered a basic set of mathematical concepts and their presentations to be handled by the notation selection tool in its default configuration. To systematize the scope of the mathematical choices, we have organized them into categories, usually corresponding to major domains of mathematics of their large subsets. We have defined the following categories: *Arithmetic*, *Combinatorics*, *Calculus*, *Linear Algebra*, *Set Theory*, *Trigonometry*. We populated each of these categories with mathematical concepts. In the category "Arithmetic" we placed operations for "multiplication", "division" and "continued fractions". The category "Calculus" originally contained operations for ordinary and partial differentiation and integration. Later as we have added more special cases for integration (limit positions, term order, *etc*), we decided to split this category into *Derivatives* and *Integrals*. We also split *Intervals* and *Logarithms* into two separate categories and created one for *Power and Root*. Each of the items within a category was then given a set of alternative notations.

Another way of systematizing mathematical domains would be to take a set of currently existing OpenMath Content Dictionaries (CDs) and create categories for each of them. Then the set of `OMS` symbols within each CD would define the content of each category. In the experimental design for our tool we decided to stay with our simplified set of mathematical operations.

3.2 The configuration file

Even though we tried to include a number of popular mathematical concepts in the initial configuration of our Notation Selection Tool, we could not guarantee that default settings will satisfy an average user. We therefore organized our software in a such a way that the end user can introduce other mathematical objects and their own notations. This is done by updating the *configuration file* that is used to initialize the Notation Selection Tool.

The configuration file contains a database of concepts, organized by category, and specifies alternative notations. We found an XML format suitable to store this file. (In particular this simplified storing template XSLT rules.) In this way the structure of the configuration file can naturally represent the classification introduced in previous section. The top-level organization of the configuration file is illustrated by the excerpt shown in Figure 4.

Each mathematical category is represented in the configuration file by the element `catalog`. Supported concepts and operations from each category have `item` elements nested inside of `catalog`. Each item is given by an OpenMath expression and is assigned a list of *notation choices* given within `choicelist` element. For instance, the operation of differentiation may be encoded as the OpenMath symbol `diff` from the Content Dictionary `weylalgebra1` and can be given the notation choices: f_x, f'_x, $\frac{\mathrm{d}f}{\mathrm{d}x}$ or $\mathrm{D}_x f$, etc.

Each notation choice, in turn, binds an example of the notation appearance (presented by a reference to an image file) to a set of converters (given by their XSLT templates). Each template defines the rules for *transformation of mathematical content to its presentation* according to the notational choice. Figure 5 provides an example of such rules to generate notations for the inverse sine function.

```
<catalog>
  <name> LINEAR ALGEBRA </name>
  <itemlist>
    <item>
      <keyword> INNER PRODUCT </keyword>
      <content>
          <!-- OpenMath encoding for mathematical concept INNER PRODUCT -->
      </content>
      <choicelist>
        <choice>
          <!-- The first notation choice for INNER PRODUCT -->
          <image src = "inn_prod1.gif"/>
          <keyvalue> 1 </keyvalue>
          <presentation>
             <converter input = "Content MathML" output="Presentation MathML">
             ... <!-- XSLT template for Content MathML to Presentation MathML -->
             </converter>

             <converter input = "OpenMath" output="LaTeX">
             ... <!-- XSLT template for OpenMath to LaTeX for this notation -->
             </converter>

             ... <!-- other possible converters -->
          </presentation>
        </choice>
        ... <!-- Other notation choices for INNER PRODUCT -->
      </choicelist>
    </item>
    ... <!-- Other items within LINEAR ALGEBRA -->
  </itemlist>
</catalog>
```

Figure 4: Configuration file excerpt

3.3 Stylesheet generation

After the user of the Notation Selection Tool selects a set of preferred notations, XSLT templates associated with each of the chosen notations are inserted into a new stylesheet. This new stylesheet combines the selected rules with the initial stylesheet supplied with our software.

The XSLT templates output by the Notation Selection Tool override the rules for the same input patterns in the base stylesheet. This is ensured by assigning higher priorities to the XSLT templates than to the corresponding templates in general stylesheet. These priorities are given by a value of the `priority` attribute appearing in each template. The generated stylesheet can then be further used within the tool or used independently for conversion of XML objects from encoding mathematical content to a presentational form (see Figure 6). We note specifically that the generated stylesheet may be used to translate to presentation form alone or it can be used in a "content-faithful" manner to produce output that retains the original semantics.

By default, the Notation Selection Tool produces a stylesheet to convert from Content MathML to Presentation MathML. The configuration file also may define templates for

```
<item>
   <title> Inverse sine of x </title>
   <keyword> ARCSINE </keyword>
   <content>
     <om:OMS cd="transc1" name="arcsin"/>
   </content>
   <choicelist>
    <!-- ******** ARCSINE CHOICE 1 ********************** -->
      <choice>
        <image src = "arcsin1.gif"/>
        <keyvalue> 1 </keyvalue>
        <presentation input="CMathML" output = "PMathML">
           <xsl:template match = "mml:arcsin" mode = "trigonometry" priority="100">
              <msup>
                 <mo>sin</mo>
                 <mn>-1</mn>
              </msup>
          </xsl:template>
        </presentation>
        <presentation input="OpenMath" output = "LaTeX">
           <xsl:template match = "om:OMS[@cd='transc1' and @name='arcsin']" priority="100">
              \sin^{-1}
          </xsl:template>
        </presentation>
      </choice>
    <!-- ******** ARCSINE CHOICE 2 ********************** -->
      <choice>
        <image src = "arcsin2.gif"/>
        <keyvalue> 2 </keyvalue>
        <presentation input="CMathML" output = "PMathML">
           <xsl:template match = "mml:arcsin" mode = "trigonometry" priority="100">
              <mo>arcsin</mo>
           </xsl:template>
        </presentation>
        <presentation input="OpenMath" output = "LaTeX">
           <xsl:template match = "om:OMS[@cd='transc1' and @name='arcsin']" priority="100">
              \mathop{arcsin}
           </xsl:template>
        </presentation>
      </choice>
    <!-- ******** ARCSINE CHOICE 3 ********************** -->
      <choice>
         <image src = "arcsin3.gif"/>
         <keyvalue> 3 </keyvalue>
         <presentation input="CMathML" output = "PMathML">
            <xsl:template match = "mml:arcsin" mode = "trigonometry" priority="100">
               <mo>asin</mo>
            </xsl:template>
         </presentation>
         <presentation input="OpenMath" output = "LaTeX">
           <xsl:template match = "om:OMS[@cd='transc1' and @name='arcsin']" priority="100">
              \asin
           </xsl:template>
         </presentation>
     </choice>
  </choicelist>
</item>
```

Figure 5: Example of rules to generate notations for the inverse sine function

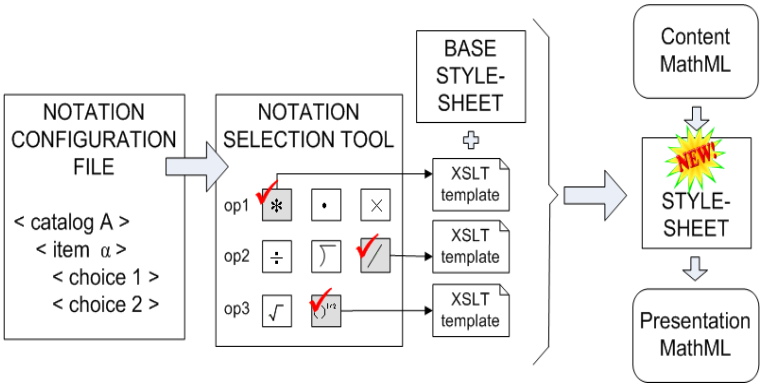

Figure 6: Notation Selection Tool in action

other types of conversion. This is indicated by the values of attributes `input` and `output`. If another type of translation is chosen, an appropriate base stylesheet must be used. The initial software kit for the Notation Selection Tool provides a base stylesheet and configuration file only for Content MathML to Presentation MathML translation. We address the question of supporting other types of conversion in Section 3.5

3.4 Extensible design

A common problem with software packages it that once they are released they do not allow users sufficient control over their behaviour. In particular, mathematical software packages often have many built-in underlying assumptions that permeate their handling of mathematical expressions. In contrast, the whole point of providing notation selection is to allows users flexibility. Ensuring flexibility and extensibility of our Notation Selection Tool has therefore been been one of our guiding principles.

To provide the desired flexibility we have used the idea of an editable configuration file. This allows the user to introduce new notations for existing math concepts by adding to this file. In particular, the set of catalogs and their mathematical content may be changed.

In the same way, new mathematical concepts can be created in existing settings. Doing this entails introducing notational choices, backed by stylesheet tools, to act as targets of those choices. For example, binomial coefficients and continued fractions are defined neither in Content MathML nor in Presentation MathML. They can be presented, however, through the use of annotated `<csymbol>` elements or by stylesheet templates for newly-defined elements. Annotated `<csymbol>` elements use a `definitionURL` attribute to reference some agreed-upon definition. (There is not necessarily any data *at* the URL location.)

```
<apply>
   <csymbol definitionURL="http://orcca.on.ca/MathML/newelement.html#binom">
   <ci> n </ci>
   <ci> m </ci>
</apply>
```

To introduce new elements in Presentation MathML we can define new XML structures as extensions of MathML. The elements of this extended MathML may be identified using a specific namespace. Thus, different notations for binomial coefficients in Presentation MathML can be defined using an XML Schema or DTD:

```
<!ELEMENT mmlx:binom( math:mi, math:mi)>
<!ELEMENT mmlx:choose( math:mi, math:mi)>
```

To define the rendering for these new elements, one can define the following XSLT transformation rules:

```
<xsl:template match = "apply/mmlx:binom[position()=1][count(child::*)=2]">
  <msupsub>
    <mfrac thickness="0ex">
      <mo> C </mo>
      <xsl:for-each select = 'mmlx:binom/child::*'>
         <xsl:copy-of select='.'/>
      </xsl:for-each>
    </mfrac>
  </msupsub>
</xsl:template>

<xsl:template match = "apply/mmlx:choose[position()=1][count(child::*)=2]">
  <mfenced>
    <mfrac thickness="0ex">
      <xsl:for-each select = 'mmlx:binom/child::*'>
        <xsl:copy-of select='.'/>
      </xsl:for-each>
    </mfrac>
  </mfenced>
</xsl:template>
```

The first template provides the extended Presentation MathML element `mmlx:binom` with the notation C^{arg1}_{arg2}. The second generates the presentation $\binom{arg1}{arg2}$ for the element `mmlx:choose`.

The same approach allows one to set preferred renderings for OpenMath CDs with Presentation MathML, as not every OpenMath symbol has corresponding elements in Presentation MathML. We can apply this technique to introduce new elements to Presentation MathML. The new elements may then be used in XSLT templates within the notation configuration file. In this way, OpenMath symbols can be mapped directly to the set of extended MathML elements. This is beneficial in two ways: First it makes the correspondence between OpenMath entities and their presentation more natural. Second this ease the creation of XSLT templates for the configuration file and allows a more compact format for the configuration file itself.

3.5 Notation selection in combination with additional transformations

So far, we have concentrated on the use of our Notation Selection Tool to build MathML transformation stylesheets. The tool is designed, however, to drive transformations between a wider range of data formats, as shown in Figure 7. The common characteristics of these

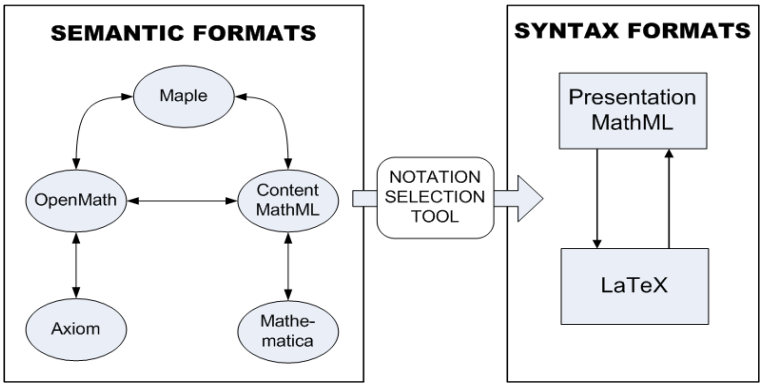

Figure 7: Mathematical data format translations implemented

conversions is that they typically take objects from high-level semantic views to lower-level renderings.

One way to enable these transformations is to introduce new entries in the configuration file. This can be done by placing additional XSLT templates within the `<presentation>` element and specifying values of the `input` and `output` attributes to indicate the type of translation. A second approach is to keep the variety of XSLT templates in the configuration file to the minimum possible, *i.e.* to support only one type of conversion, for instance Content MathML to Presentation MathML. For the other formats, external conversion tools can be used. Our group at the Ontario Research Centre for Computer Algebra has developed various data format translation techniques supporting conversion[1], shown in Figure 7.

OpenMath ↔ Content MathML The conversion between these two XML-based formats is performed natively by XSLT stylesheet transformations. We have designed the stylesheets for a 2-step transformation in both directions. Conversion from OpenMath to Content MathML may either elect to generate the built-in MathML operations, when appropriate, or to give everything in a general format using `csymbol`s. In this direction our converter supports arbitrary OpenMath CDs. Conversion in the other direction, from Content MathML to OpenMath, understands the standard OpenMath CDs.

OpenMath ↔ Maple The translator between Maple objects and OpenMath expressions is organized as an engine driven by a set of mapping files. These files describe the correspondence between patterns of mathematical structures represented by both formats. The scope of the mathematical content that the converter can handle is determined by the set of mapping files it has loaded. This allows us to configure the converter vocabulary, which

[1]The conversions {Maple, Mathematica} → Content MathML and Axiom → OpenMath are internally supported in the corresponding systems. The conversions {Maple,Mathematica} → Presentation MathML and {Maple,Mathematica,Axiom} → LaTeX are also supported internally, but these conversions use only default notation, defined by the systems.

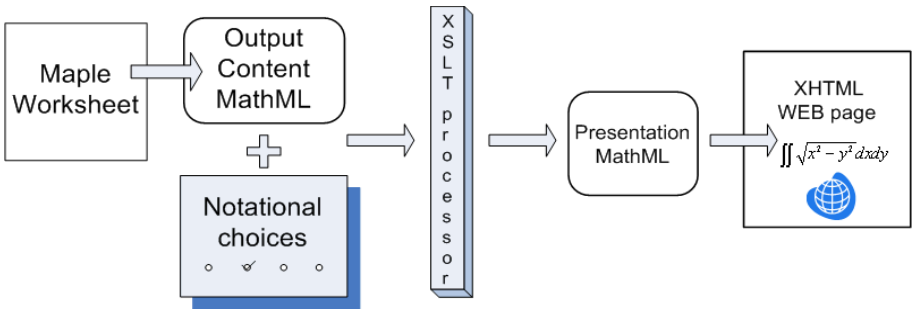

Figure 8: Using notation selection to produce web pages with mathematical content

is necessary to support OpenMath's extensible nature. It is convenient for every mapping file to correspond to one OpenMath Content Dictionary. It is used to drive both directions of the translation. The translator engine itself is implemented in Maple, which makes the conversion tool native to the environment of the computer algebra system, while remaining configurable by the user.

Presentation MathML ↔ LaTeX Even though the most common approach to convert from MathML to TeX is, again, to use XSLT stylesheets, we chose *not* to use this method. The reason is that we wanted our converter to be symmetric to our TeX to MathML converter and for both of them to be configured by one set of bidirectional mappings. Since TeX is not an XML-based format, we could not use XSLT to translate from TeX to MathML. Therefore, we have developed two Java packages, one for each direction of conversion to and from TeX. Both programs use the *same* mapping file to set the correspondence between MathML and LaTeX syntactic patterns. This mapping file is similar to the configuration file used by the Notation Selection Tool and can be edited by the users of the converter. This approach allows flexibility and extensibility of the tools for these conversions. The other benefit of using mapping files is preserving high-level semantics in transforming between TeX macros and MathML extension elements. The on-line version of the LaTeX ↔ MathML translator is available at `http://www.orcca.on.ca/MathML/texmml`. More details on the MathML to LaTeX converter are given in [21].

4 Applications of Notation Selection

One of the most common applications for the Notation Selection Tool is support for multiple formats in mathematical documents. One can imagine a likely scenario in which a user may wish to export the computation results from a computer algebra system and to place them on a web page within lecture notes or on-line publication. The rendering of mathematical content then may be specified by a choice of notation set trough the Notation Selection Tool (see Figure 8).

Another area of application for our Notation Selection Tool is that of mathematical education, where students require a high degree of notational consistency within a syllabus. Notation selection facilities can allow an instructor to re-use material with different notational conventions from one course to another. In distance learning, students might prefer to see mathematical expressions in the format of their locality, so our tool could be used to choose these preferences.

We describe below a few of our on-going investigations which involve our Notation Selection Tool.

4.1 From syntax to semantics

We have discussed how, at a syntactic level, mathematical notation can be highly ambiguous. This may not be apparent to most mathematical readers, as they will be using a great deal of semantic contextual information and moreover any particular document will make use of only a few areas of mathematics. We suggest, therefore, that the general semantic attribution of pure syntactic mathematical expressions is a suitable well-defined problem in the area of artificial intelligence.

From the point of view of mathematical software, however, we do not necessarily require handling of general expressions in every possible context. It may be perfectly acceptable to require the user, or application, to specify explicitly the mathematical context in which the expressions occur. In this case, with the mathematical domain suitably narrowed, semantic attribution of expressions becomes much more tractable.

We anticipate that the use of tools, such as our Notation Selection Tool, will be useful in this setting. By selecting a set of notational preferences, the user defines a space of admissible expressions together with their interpretation. Clearly, not all possible combinations of selected notations will lead to unique interpretation of all expressions. However, once a selection of preferences is made, it is possible to determine in advance the areas where ambiguities will arise (see Figure 9).

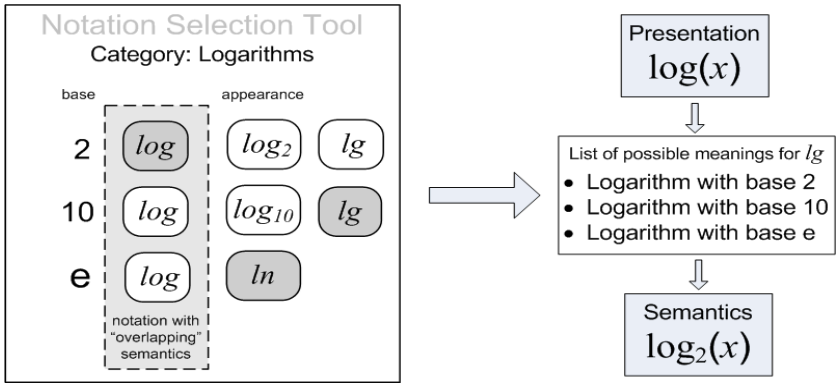

Figure 9: Notation selection in syntax disambiguation

4.2 Notation selection in mathematical handwriting recognition

Input to mathematical software comes in the form of expressions in various parsed linear syntaxes, expressions built using interactive expression editors, in MathML or TEX. We are currently engaged in a project that attempts to add handwritten mathematics as an input method[18]. Handwritten input presents an interesting and difficult case, because it relies both on accurate single character recognition within a large set of mathematical symbols and on structural analysis of 2-dimensional layout. We see an opportunity for the Notation Selection Tool to assist in this process, both for handwritten input to computer algebra systems (see Figure 10) and other applications (Figure 11).

As a high-level design objective, our expression recognizer must support various notations in handwritten input. This leads to problems, however, when notations are ambiguous or very similar. Attempting to disambiguate between many such choices would be both time consuming and reduce recognizer accuracy. To narrow down the set of expression models without forcing the user's choice of notation, we can ask the user to use our Notation Selection Tool to specify preferred notations.

Eliminating un-used notations has a first benefit of reducing the set of candidates in single character analysis. For example, if a user has selected $\sim$ instead of $\propto$ for proportionality, then $\propto$ can be eliminated from character candidate lists. This will lead to more accurate recognition of similar characters, such as α and ∞.

Once character candidates have been generated and ranked, it is necessary to select the most likely choice while taking into account expression context analysis. Notation selection can be used here to activate or deactivate various rules in the structural recognition process. For example, if French-style interval notation is selected, then parentheses and brackets do not have to match.

At present, our recognizer uses a combination of single character analysis and character prediction. The prediction is based on writing order Markov chains compiled from the analysis of some 20,000 mathematical documents. The source for these mathematical documents, however, is in TEX form and there is no distinction made in the analysis between different meanings for the same notation [19][20]. We have not yet explored whether notation selection applied to the document database would lead to more better prediction. A second use of notation selection that we have not yet explored for mathematical handwriting recognition is the use of correlated notations. For example, if the notation $\int \mathrm{d}t f(t)$ is selected for integration, then it would be reasonable to assign higher likelihoods to other notations from physics.

4.3 Notation selection in mathematical knowledge management

One of the areas in mathematical knowledge management is to organize or classify entries in mathematical knowledge bases. Such categories are well defined and supported by a number of systems and databases including Mizar [13], MBase [10], the NIST Digital Library of Mathematical Functions [12], OpenMath Content Dictionaries [3], to name a few. The work [15] is also interesting in this regard.

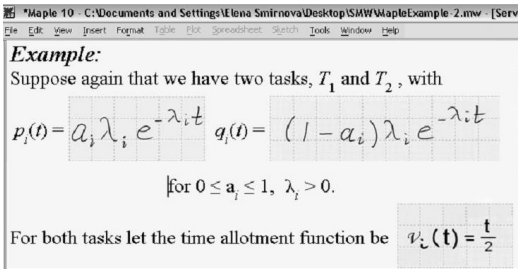

Figure 10: Pen-Enabled input in Maple

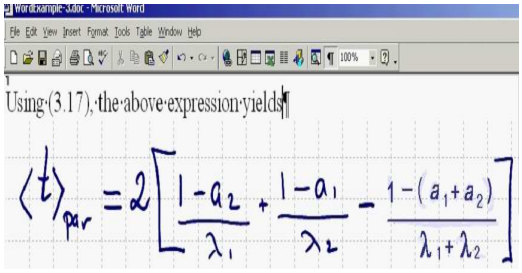

Figure 11: Pen-Enabled input in Word

A goal with these categorizing tools is to avoid storing duplicated information. Each knowledge base will choose its representations for particular functions, and organize information about them accordingly. For example, if information about the hyperbolic sine function uses the function $\sinh(z)$ then users from Eastern Europe will find nothing about $\mathrm{sh}(z)$ because there is no such entry.

One possible solution is to include multiple function names in the database, but this is fraught with problems. If, for example, rules are modified for $\arcsin(z)$ but not for $\sin^{-1}(z)$, is this an intentional subtle distinction between inverse functions with different branch cuts or is it a bug? Should $\arcsin(x) - \sin^{-1}(z)$ simplify to zero? It is clearly preferable to use a notation selection tool as a component to the front end to such mathematical knowledge bases.

4.4 Notation selection in computer algebra systems

Most of the issues we have raised so far is directly applicable to the input and output of computer algebra systems. The ability adapt the interpretation of input forms and to select the possible forms of output has obvious value in making these systems more accessible to different communities.

There are certain practical considerations, however, that impact the successful adoption of this strategy in current systems. The most widely used mathematical software packages are designed with the input and output systems separate from one another. In this situation, to support proper notation selection, it is necessary to apply transformations to *both* the input and output expressions. Difficulties arise when the input and output transformations are inconsistent. This makes it difficult to effectively modify most existing computer algebra systems to support notation selection coherently: both the input and output systems are typically sizable, complex software subsystems, and obtaining consistency between them is challenging. We have experienced this situation with our suite of transformation tools to convert between various mathematical formats, and in particular converting between TEX and MathML[2]. In this situation we have found a useful approach is to specify a set of bi-directional transformation rules, used independently by the translators for both directions.

Beyond the question of consistent handling of input and output notation selection, computer algebra systems present an additional problem: They generate new mathematical expressions as part of their operation and these expressions may contain functions or other symbols not anticipated by the user. For example, in the solution of differential equations a wide variety of special functions may be generated, some of which might be unknown to the user of the system. It is unreasonable to disallow the user to employ notations that denote standard functions. There are so many that *could* occur, but only a few actually *will* in any particular setting. On the other hand, always using non-standard names for standard functions (e.g. `BesselJ[0](z)` instead of $J_0(z)$) leads to bulky expressions that may not be used directly in other settings. It is therefore necessary to properly manage the interactions between notations deliberately set by the user and names that may occur in generated expressions.

5 Conclusion

We have observed that a wide range of software, including document processors and computer algebra systems confuse *mathematics* with *notation*. By forcing too early the choice of notation, flexibility is lost and work is restricted to a too narrow context. We have shown how software tools can be used to translate meaningful mathematical constructs both to and from a wide range of notations. This can be used to allow mathematical documents and computational worksheets to be deployed in a wide range of settings, with notation customized by country, mathematical field, level of sophistication, or other criterion.

We have described the rationale, design and implementation of a Notation Selection Tool that allows the interactive construction of stylesheets to convert between mathematical formats. The current state of the art is presently implemented for transforming mathematical data encoded in Content MathML to Presentation MathML. When used in conjunction with additional conversion tools the following translations are also available: {Content MathML, OpenMath, Maple, Axiom, Mathematica} → {Presentation MathML, LaTeX}.

We have discussed a number of settings where notation selection tools can be useful in defining a restricted domain of discourse. This disambiguation can be useful in settings that require some semantic treatment of a wide range of mathematical subjects. In particularly, we have discussed notation selection in the context of computer algebra, in improving accuracy in pen-based mathematical interfaces and in mathematical knowledge management applications.

We see an interesting future project in incorporating a flexible Notation Selection Tool in major computer algebra systems, such as Maple. For maximum utility, this must work bi-directionally: to select notations to be used for both input and output. Mathematical structure editors can allow us to avoid the technical parsing problems that would be introduced through user-defined linear input syntax.

References

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

[2] R. Ausbrooks at al. Mathematical Markup Language (MathML) Version 2.0 (Second Edition), World Wide Web Consortium Recommendation 21 October 2003. http://www.w3.org/TR/2003/REC-MathML2-20031021

[3] S. Buswell *et al.*, The OpenMath Standard 2.0. 2004 http://www.openmath.org/cocoon/openmath/standard/om20/index.html

[4] James Clark, World Wide Web Consortium Recommendation, XSL Transformations (XSLT), December 1999. http://www.w3.org/TR/XSLT

[5] James H. Davenport, Mathematical Knowledge Representation, Electronic Proceedings of the First International Workshop on Mathematical Knowledge Management: MKM 2001 RISC, A-4232 Schloss Hagenberg, September 24-26, 2001.

[6] Jason Harris, Advanced Notations in Mathematica, Proceedings of the International Symposium on Symbolic and Algebraic Computation, 2000.

[7] Sandy Huerter, Igor Rodionov and Stephen M. Watt, Content-Faithful Transformations for MathML, Proc. International Conference on MathML and Math on the Web, (MathML 2002), http://www.mathmlconference.org/2002, June 28-30 2002, Chicago USA.

[8] Java Server Pages, http://java.sun.com/products/jsp/

[9] Java Swing, Robert Eckstein, Marc Loy and Dave Wood, 1998, O'Reilly.

[10] M. Kohlhase, A. Franke. MBase: Representing Knowledge and Context for the Integration of Mathematical Software Systems. Journal of Symbolic Computation 23:4 (2001), pp. 365 - 402.

[11] Dicheng Liu, A Notation Selection Tools for MathML stylesheets, MSc Project University of Western Ontario, 2001

[12] D.W. Lozier, NIST Digital Library of Mathematical Functions. In Annals of Mathematics and Artifcial Intelligence, vol. 38, No. 1-3, May 2003. Eds. B. Buchberger, G. Gonnet, M. Hazewinkel. Kluwer Adacemic Publishers, ISSN 1012-2443.

[13] The Mizar System, http://mizar.uwb.edu.pl/system/

[14] W.A. Naylor and Stephen M. Watt, Meta-Stylesheets for the Conversion of Mathematical Documents into Multiple Forms, Annals of Mathematics and Artificial Intelligence, Vol. 38, pp. 3-25, 2003.

[15] Florina Piroi and Bruno Buchberger, An Environment for Building Mathematical Knowledge Libraries. Proc. Fourth International Conference on Mathematical Knowledge Management, (MKM 2005), July 15-17 2005, Bremen Germany, Springer Verlag

[16] Igor Rodionov and Stephen M. Watt, Content-Faithful Stylesheets for MathML, Ontario Research Centre for Computer Algebra, University of Western Ontario, Research Report TR-00-14, 2000.

[17] Elena Smirnova and Stephen M. Watt, An Approach to Mathematical Notation Selection, North American Mathematical Knowledge Management Workshop , (NA-MKM 2004), 2004, Phoenix Arizona.

[18] Elena Smirnova and Stephen M. Watt, A Context for Pen-Based Computing, pp. 409-422, Proc. Maple Conference 2005, July 17-21 2005, Waterloo Canada, Maplesoft.

[19] Clare M. So and Stephen M. Watt, Determining Empirical Properties of Mathematical Expression Use , Proc. Fourth International Conference on Mathematical Knowledge Management, (MKM 2005), July 15-17 2005, Bremen Germany, Springer Verlag

[20] Clare M. So, An Analysis of Mathematical Expressions Used in Practice, MSc Thesis, University of Western Ontario, 2005

[21] Stephen M. Watt, Exploiting Implicit Mathematical Semantics in Conversion between TEX and MathML, Proc. Internet Accessible Mathematical Communication, (IAMC 2002)

[22] XML specification: http://www.w3.org/XML.

Elena Smirnova
Ontario Research Centre for Computer Algebra
University of Western Ontario
elena@orcca.on.ca
www.orcca.on.ca/MathML/elena.html

Stephen M. Watt
Ontario Research Centre for Computer Algebra
University of Western Ontario
watt@orcca.on.ca
http://www.orcca.on.ca/~watt