Generating TEX from mathematical content with respect to notational settings

Elena Smirnova
Ontario Research Centre for Computer Algebra
The University of Western Ontario
London, ON, N6A 5B7, Canada
`elena (at) orcca dot on dot ca`
`http://www.orcca.on.ca/MathML/elena.html`

Stephen M. Watt
Department of Computer Science
The University of Western Ontario
London, ON, N6A 5B7, Canada
`watt (at) csd dot uwo dot ca`
`http://www.csd.uwo.ca/~watt.html`

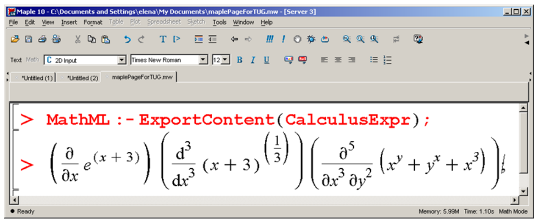
Abstract

We describe how to obtain client-preferred notations in TEX generated from the output of mathematical software environments. Our approach is based on the fact that most packages can produce MathML or other XML-based formats for mathematical content. Generating TEX from these allows notational choices to be applied during the translation process. The particular choices of notation can be made either at the time TEX is generated or later, by the use of TEX macros. We show how this approach may be applied to the generation of TEX from both *presentationally*- and *conceptually*-oriented mathematical content and how MathML may be used in the process. Our implementation conserves the implicit high-level semantics of macro use in both TEX and MathML. Since the expressions generated by mathematical software may be quite lengthy, we also discuss issues that arise in line-breaking.

1 Introduction

Most mathematical software systems allow one to export expressions in TEX format. For many mathematical ideas there are several choices of notation, and typically the TEX formulae generated by software systems use the notational conventions selected by the system designers. These choices might be quite different from what would be selected by the client of the package, if a choice were offered.

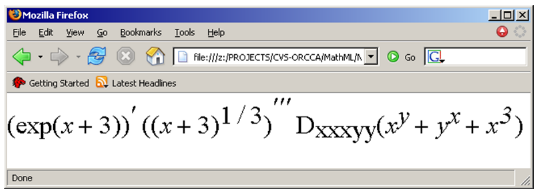
We are interested in the problem of generating TEX that respects the notational conventions that are preferred by the client. To see the difference between a default and a customized rendering of an expression, compare the formulae of Figure 1a and Figure 1b. In most cases it is not possible for a user to obtain TEX output that uses their preferred notation. If the user of a computer algebra system or other mathematical software package wishes to publish the results of a computation, he or she must either accept the presentation offered by the system, or rewrite the TEX content. In this paper we present an alternative approach, based on adding notation preferences to mathematical TEX converters.



```
> MathML :- ExportContent(CalculusExpr);
```

$$\left(\frac{\partial}{\partial x} e^{(x+3)}\right)\left(\frac{d^3}{dx^3}(x+3)^{\left(\frac{1}{3}\right)}\right)\left(\frac{\partial^5}{\partial x^3\,\partial y^2}\left(x^y+y^x+x^3\right)\right)$$

(a) Default rendering by Maple



$$(\exp(x+3))'\,((x+3)^{1/3})'''\,\mathrm{D}_{xxxyy}(x^y+y^x+x^3)$$

(b) Customized rendering via Mozilla

Figure 1: Two renderings of the same expression.

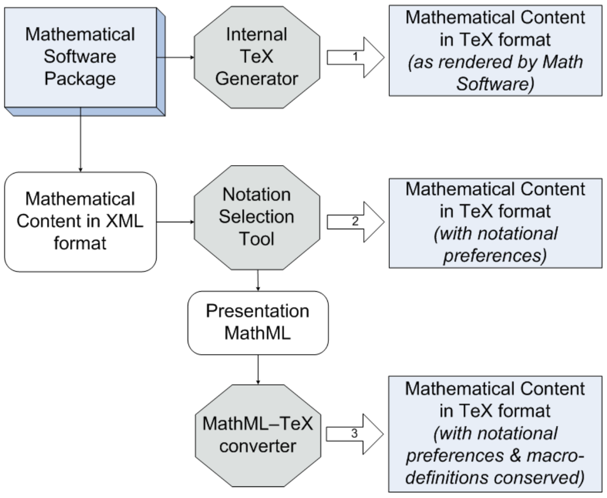

Mathematical Software Package
Internal TeX Generator
1
Mathematical Content in TeX format (as rendered by Math Software)
Mathematical Content in XML format
Notation Selection Tool
2
Mathematical Content in TeX format (with notational preferences)
Presentation MathML
MathML–TeX converter
3
Mathematical Content in TeX format (with notational preferences & macro-definitions conserved)

Figure 2: Three ways to generate TEX from a mathematical software system

There are several ways in which TEX for mathematical expressions may be generated from a software package such as Maple [1], Mathematica [2], Axiom [3], Aldor [4], *etc.* One approach is to make a direct translation to TEX from the internal expression representation of the software system. Exporting it to TEX directly from the system will then produce package-specific presentation of this content (Figure 2, arrow 1). Another approach may be taken if the mathematical object has been created in a web-oriented mathematical environment, such as with the MONET web-services [5, 6]. In this situation the expression will most likely be encoded using some XML-based standard, such as OpenMath [7] or Content MathML [8]. These formats are supported by most computer algebra systems as well as many web browsers and other packages. The objective of this paper is to demonstrate a flexible technique to generate of TEX presentation from these XML-based, semantically-oriented formats.

We explore two manners of producing TEX from XML formats: The first is to do so directly from mathematical content, taking into account notational settings (Figure 2, arrow 2). The second approach is to use a two-stage conversion (Figure 2, arrow 3). In this case, MathML — combined with elements using some extended set of tags — is first generated from the mathematical content. This extended MathML is then translated into TEX with corresponding TEX macros. In this case, high-level mathematical constructs may be mapped directly to TEX macros. This ensures that the semantics of the original expression are conserved in the output TEX content.

This paper is organized as follows: Section 2 describes how presentation of mathematical content can be customized using a Notation Selection Tool. Section 3 describes a MathML to TEX translator that is used in multi-stage conversion. Section 4 provides some details of how line breaking is achieved in the generated TEX. Section 5 presents our conclusions and outlines some possible directions for future work in this area.

2 Generating presentation from content using notational preferences

Anyone with more than a passing familiarity with the subject understands that there is no universal notation for mathematics. There are mathematical concepts for which there are several different notations, and there are notations for which there are several different mathematical concepts. Figure 3 shows how the choice of different notation can affect the appearance of mathematical content. This choice of notation is certainly the major determining factor in how an expression will appear.

2.1 A notation selection tool

In earlier work we described a Notation Selection Tool [9, 10] designed to control conversion of mathematical expressions in XML format. The original purpose of this tool was to provide a graphical user interface (Figure 4) for notation selection. The tool generates an XSLT stylesheet used to transform Content MathML to Presentation MathML using the desired notational conventions. The stylesheet and the generated Presentation MathML are determined by the user's choice of notation settings.

2.2 Extending the tool for direct conversion to TEX

In the present work we use a key feature of the Notation Selection Tool: Its extensible design allows one to add new translation directions without modifying the software implementation. In particular, we can add direct conversion of OpenMath and Content MathML to TEX.

We described in [9] how the Notation Selection Tool is initialized by a configuration file. This file is the only component in the design that provides information about the mathematical concepts that the converter can handle. It also stores the transformation rules to be applied for the selected notations.

Because all knowledge of the MathML conversion is contained in this configuration file, it may also be used to specify conversions involving other XML formats for input and other XML or text formats for output. Updating the configuration file

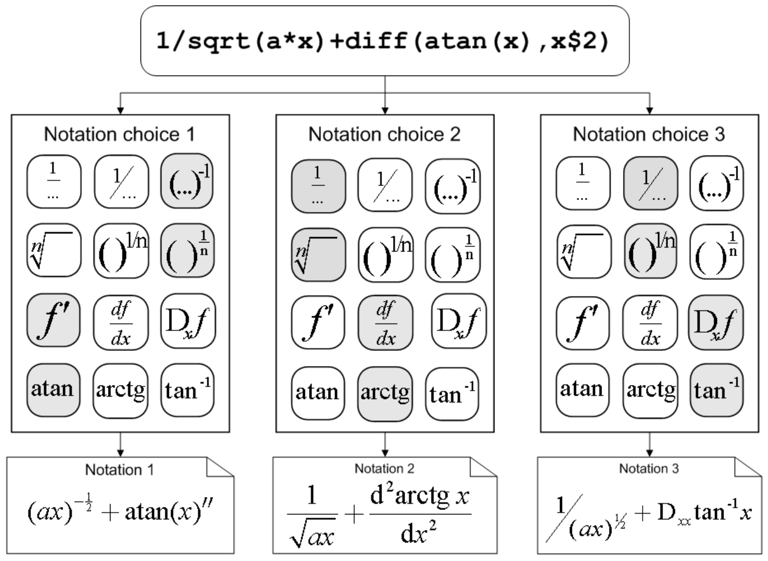

1/sqrt(a*x)+diff(atan(x),x$2)
Notation choice 1
Notation choice 2
Notation choice 3
Notation 1
$(ax)^{-\frac{1}{2}} + \mathrm{atan}(x)''$
Notation 2
$\frac{1}{\sqrt{ax}} + \frac{\mathrm{d}^2 \mathrm{arctg}\, x}{\mathrm{d}x^2}$
Notation 3
$1/(ax)^{1/2} + \mathrm{D}_{xx} \tan^{-1} x$

Figure 3: Generating different notations for the same mathematical content

with new conversion rules, such as with OpenMath as source and TEX as target, allows the Notation Selection Tool to perform new conversions according to desired rules:

```
<catalog>
  <name> CALCULUS </name>
  <itemlist>
    <item>
      <keyword> PARTIAL DERIVATIVE </keyword>
      <content>
          OpenMath encoding for mathematical
          concept PARTIAL DERIVATIVE
      </content>
      <choicelist>
        <choice>
          <!-- The first notation choice for  -->
          <image src = "pd_1.gif"/>
          <keyvalue> 1 </keyvalue>
          <presentation>
             <converter input = "OpenMath" output="LaTeX">
                XSLT template for OpenMath
                to LaTeX for this notation
             </converter>
             ...
          </presentation>
        </choice>
        ...
      </choicelist>
    </item>
    ...
  </itemlist>
</catalog>
```

If the user selects the notation D_x for partial differentiation and f' for ordinary differentiation, then, instead of obtaining the default output, the Maple expression shown in Figure 1a will be converted to

```
$${{\left(\mathop{exp}{\left({x+3}\right)}\right)}^\prime}\,
  {{({{\left({x+3}\right)}^{1/3}})}^{\prime\prime\prime}}\,
  {{\mathrm D}_{xxxyy}\left({{x^y}+{y^x}+{x^3}}\right)}$$,
```

which renders as

$$(exp\,(x+3))'\,((x+3)^{1/3})'''\,\mathrm{D}_{xxxyy}\left(x^y+y^x+x^3\right).$$

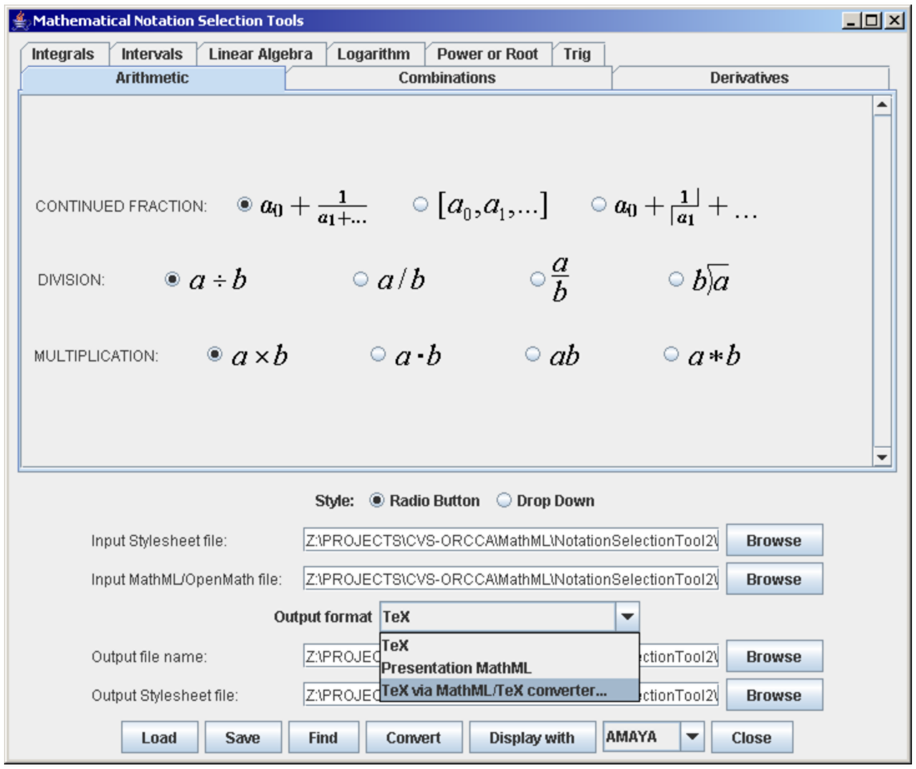
The configuration file may define more than one output format, for example both LaTeX and MathML. In this case, the conversion rules are selected according to the target format specified by the user (see Figure 4). This approach is a special case of that described in [11] for the conversion of mathematical documents into multiple forms.

2.3 The notation selection tool as a front-end in multi-stage conversion

We have discussed extension of the Notation Selection Tool by modifying the configuration file to provide new conversions *directly* to TEX. We can, instead, do the translation in a *series of stages* to increase flexibility.

We can use the Notation Selection Tool as an intermediate translation stage generating MathML. This MathML can then be further processed to produce TEX. This multi-stage mode of generating TEX from Content MathML or OpenMath can be specified by a menu selection in the Notation Selection Tool. In this mode, the Notation Selection Tool first generates MathML and then passes it to our configurable MathML to TEX converter [12].

This multi-stage process allows the Notation Selection Tool to generate extended markup (*i.e.* MathML and other XML) that may then later be transformed. To do this, the desired extended markup is placed in the output rules of the configuration file. This extended markup can use new tags to capture the semantics of new mathematical concepts or complex combinations of OpenMath or MathML constructs. For example, one might use a new XML



CONTINUED FRACTION: [✓ $a_0+\frac{1}{a_1+\dots}$] $[a_0,a_1,\dots]$ $a_0+\frac{1|}{|a_1}+\dots$

DIVISION: [✓ $a \div b$] a/b $\frac{a}{b}$ $b\overline{)a}$

MULTIPLICATION: [✓ $a \times b$] $a \cdot b$ ab $a*b$

Style: [✓ Radio Button] Drop Down

Input Stylesheet file: Z:\PROJECTS\CVS-ORCCA\MathML\NotationSelectionTool2\

Input MathML/OpenMath file: Z:\PROJECTS\CVS-ORCCA\MathML\NotationSelectionTool2\

Output format: TeX

TeX
Presentation MathML
TeX via MathML/TeX converter...

Output file name: Z:\PROJEC ctionTool2\

Output Stylesheet file: Z:\PROJEC ctionTool2\

Display with: AMAYA

Figure 4: The Notation Selection Tool, stand-alone application interface

element to capture the semantics of binomial coefficients or continued fractions, which do not appear in standard MathML. In such a case, a new transformation rule can be added directly to the configuration file of the Notation Selection Tool:

```
<converter input = "Content MathML" output="LaTeX">
  <xsl:template match = "apply/mmlx:choose[position()=1]
                         [count(child::*)=2]">
   <mmlx:binomial>
        <xsl:for-each select = 'mmlx:choose/child::*'>
           <xsl:copy-of select='.'/>
        </xsl:for-each>
   </mmlx:binomial>
  </xsl:template>
</converter>
```

The output of the XSLT transformation in the above example will contain Presentation MathML extended with the new tag `mmlx:binomial`. The prefix "`mmlx`" is defined at the beginning of the generated XML, and indicates that this element belongs to a different namespace than the standard MathML elements. The resulting element can be either expanded immediately after the conversion to obtain a combination of appropriate presentation markup and semantic annotation (possibly using a `<csymbol>` element). Alternatively, it may be carried on to the next step of the translation to TEX, as described in the next section.

3 MathML to TEX conversion

In earlier work we have explored the question of conversion between TEX and MathML, using a set of bidirectional transformation rules [13, 14, 15]. Here we summarize the aspects of the converter that are used in notation selection for TEX via MathML.

3.1 Modes of conversion

In [15] we presented a MathML to TEX translator that converts expressions in MathML representation to equivalent TEX expressions. The translator supports conversion at three different levels of content granularity: (1) entire files, (2) individual expressions and (3) separate objects. The last option allows the user to manipulate stand-alone MathML and TEX-objects obtained from sources other than the usual MathML or TEX documents.

To perform the file-level conversion, the translator processes an entire MathML file and produces a complete TEX document. The converter can also handle other XML files containing MathML and replace the MathML elements with their TEX equivalents, embedded as `CDATA` sections, as shown on Figure 5. This option is useful when our converter is used as a pre-processor for HTML to LATEX translators such as `html2latex` [16]. Thus, from a sequence of two translations we can produce TeX documents from HTML web pages containing mathematics.

3.2 Implementation

Even though the most natural choice for implementing a converter for XML-based languages such as MathML is via XSLT stylesheets, we have taken another approach. The main argument against using XSLT for this converter was our desire to provide a symmetric path for the inverse conversion from TeX to MathML. We chose to organize both of the translators based on *bidirectional* mappings between MathML and TeX constructs. Mapping rules describe the correspondence between TeX and MathML patterns in the following format:

```
<pat:template>
  <!-- TeX command with parameters -->
  <pat:tex op="\frac" params="\patVAR!{num}\patVAR!{den}"/>

  <!-- Corresponding MathML tree -->
  <pat:mml op="mfrac">
    <mfrac>
      <pat:variable name="num"/>
      <pat:variable name="den"/>
    </mfrac>
  </pat:mml>
</pat:template>
```

Templates, such as the above, are organized into mapping files. The same file can be used both for converting from TeX to MathML and *vice versa*. The converter tools are implemented in Java and read one or more mapping files as they are initialized. The converter may be run as a stand-alone application (Figure 6) or as a web service [12]. The same configuration file is used to control a transformer in the reverse direction [17], from TeX to MathML. While it is the core Java program that is carrying out the actual conversion, its *behavior* is defined by the selection of mapping files.

This approach provides the converter with the desired flexibility: none of the conversion rules are hard-coded and any of them may be updated by editing the corresponding templates in the mapping files. The consistency between the two directions of conversion is preserved, since any of the mapping files can be used by either of the converters.

New mappings can be added in a similar way. Whenever a new pattern, such as a TeX macro or XML template, is introduced, a new template can be added to a mapping file. This immediately enables the conversion using a new transformation rule.

3.3 Conserving high-level semantics in translation

In [13] and in Section 2.3 we have described how new mathematical constructs can be described with TeX macros or XSLT template definitions.

Macros are usually used as abbreviations for lengthy expressions, expressions that are particularly notable, or that appear more than once. These expressions typically have some meaning that makes them natural choices for expression by macros. When converting a mathematical document between formats, we wish to conserve whatever implicit semantics is captured by the macro markup. Expanding macros and then converting loses this information.

In [13] and [18] we showed that rather than expanding all macros to low-level formatting instructions, in many cases it is possible to map high-level markup in one setting to corresponding markup in another, thus conserving implied semantics. We may arrange that each TeX style or class file have a counterpart XSLT stylesheet for use with Presentation MathML, and each TeX macro have a corresponding XSLT template definition.

We now give a complete example: Suppose we are working with documents that involve the Lambert "W" function of two arguments k and z, denoted $W_k(z)$. It would be possible to denote this explicitly as `W_k(x)` everywhere in a TeX document, and as corresponding presentation markup in a MathML document. We prefer, however, to define a TeX macro, such as

```
\newcommand{\LambertW}[2]{W_{#1}\left( {#2} \right)}
```

and a corresponding XSLT template

```
<xsl:template match="mmlx:LambertW">
  <mrow>
     <msub>
       <mo> W </mo>
       <xsl:apply-templates
            select = 'mmlx:LambertW/child::*[1]"/>
     </msub>
     <mfenced>
       <xsl:apply-templates
            select = 'mmlx:LambertW/child::*[2]"/>
     </mfenced>
  </mrow>
</xsl:template>
```

We also add a direct mapping rule for `\LambertW` and `<mmlx:LambertW>` to one of the translator mapping files:

```
<pat:template>
  <!-- TeX command -->
  <pat:tex op="\LambertW"
         params="\patVAR!{k} \patVAR!{z}"/>
  <!-- MathML element -->
  <pat:mml op="mmxl:LambertW">
    <mmxl:LambertW>
      <pat:variable name="k"/>
      <pat:variable name="z"/>
    </mmxl:LambertW>
  </pat:mml>
</pat:template>
```

Using this mapping, a MathML expression

```
<mmxl:LambertW>
  <mn> 1 </mn>
  <mi> z </mi>
</mmxl:LambertW>
```

```
<?xml version='1.0' encoding='utf-8'?>
<html>
 <h1> XHTML + MathML </h1>
 <ol>
   <li>
    <math>       ⇐ Expression 1
      <msup>
        <mi>x</mi>
        <mn>2</mn>
      </msup>
      <mo>+</mo>
      <mn>1</mn>
    </math>
   </li>
   <li>
    <math>       ⇐ Expression 2
     <msubsup>
       <mi> A </mi>
       <mi> i </mi>
       <mi> j </mi>
     </msubsup>
    </math>
   </li>
 </ol>
</html>
```

```
<?xml version='1.0' encoding='UTF-8' ?>
<html>
 <h1> HTML + MathML </h1>
 <ol>
  <li>
   <LaTeX xmlns='orcca.on.ca'> ⇐ Expression 1
     <![CDATA[$${x^2}+1$$]]>
   </LaTeX>
  </li>
  <li>
   <LaTeX xmlns='orcca.on.ca'> ⇐ Expression 2
    <![CDATA[$${A_i^j}$$]]>
   </LaTeX>
  </li>
 </ol>
</html>
```

Figure 5: Example of conversion from XHTML with MathML to TEX

will be translated to TEX as `\LambertW{1}{z}` instead of `W_1\left(z\right)`.

This approach, converting from MathML to TEX driven by high-level rules, allows a concept-level translation of user-defined macros. This preserves mathematical semantics implied by the markup of the original expression and, most importantly for the present paper, allows user-preferred notations to be given by alternative definitions of the target TEX macros.

4 Automated line breaking

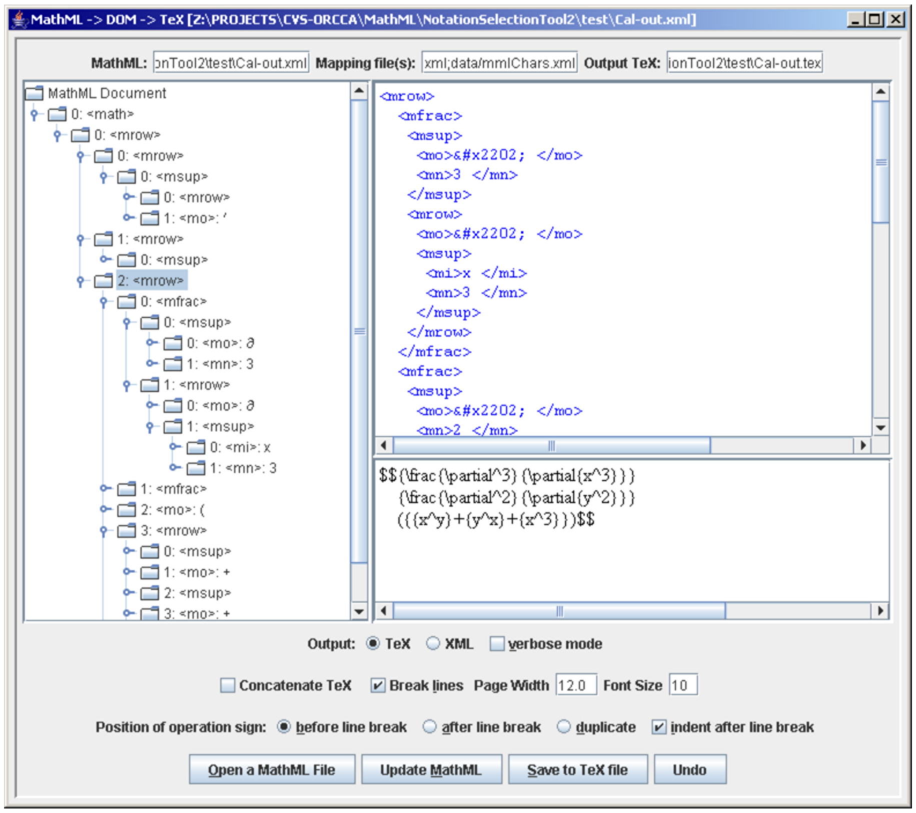
A secondary benefit of using a non-XSLT approach for conversion from MathML to TEX is that it makes automated line breaking of long expressions easier.

4.1 Motivation

In general, automated line-breaking of mathematical expressions is a complex problem and has been studied by designers of computer algebra systems for some three decades [19]. MathML browsers, such as Amaya, MathPlayer and those of the Netscape family, either have their own mechanism for line breaking in mathematical formulae, or provide a scrolling region for long expressions. TEX, however, does not natively support either of these options. Therefore long TEX formulae generated from MathML may not fit in the text area of the document. Very often composing a mathematical paper with long formulae will give results as shown in Figure 7.

Manual line breaking in TEX formulae is viable only when the size of the expression is relatively small. However, when MathML content is generated as output from a mathematical software package, or when numerous documents are to be converted, this approach is not sufficient. One solution would be to use the `breqn` package [20] to display generated TEX formulae. There are a number of reasons, however, why we have elected to provide line-breaking as part of the TEX generation:

- Generated TEX formulae can be very large, so a number of line breaking (and page breaking) issues arise that do not arise with hand-written equations. In particular, `breqn` fails to separate factors of long products with implied multiplication. We can handle these situations better than a human-oriented package such as `breqn`.
- Generated TEX formulae can be idiosyncratic, so providing our own line breaking of equations gives finer control.
- As part of the TEX generation we have already performed much of the analysis that is required for line breaking.
- As a purely practical consideration, not all TEX environments support the `breqn` package.


MathML -> DOM -> TeX [Z:\PROJECTS\CVS-ORCCA\MathML\NotationSelectionTool2\test\Cal-out.xml]
Open a MathML File
Update MathML
Save to TeX file
Undo

MathML: onTool2\test\Cal-out.xml Mapping file(s): xml;data/mmlChars.xml Output TeX: ionTool2\test\Cal-out.tex

```
MathML Document
  0: <math>
    0: <mrow>
      0: <mrow>
        0: <msup>
          0: <mrow>
          1: <mo>: ′
      1: <mrow>
        0: <msup>
      2: <mrow>
        0: <mfrac>
          0: <msup>
            0: <mo>: ∂
            1: <mn>: 3
          1: <mrow>
            0: <mo>: ∂
            1: <msup>
              0: <mi>: x
              1: <mn>: 3
        1: <mfrac>
        2: <mo>: (
        3: <mrow>
          0: <msup>
          1: <mo>: +
          2: <msup>
          3: <mo>: +
```

```
<mrow>
  <mfrac>
   <msup>
    <mo>&#x2202; </mo>
    <mn>3 </mn>
   </msup>
   <mrow>
    <mo>&#x2202; </mo>
    <msup>
     <mi>x </mi>
     <mn>3 </mn>
    </msup>
   </mrow>
  </mfrac>
  <mfrac>
   <msup>
    <mo>&#x2202; </mo>
    <mn>2 </mn>
```

```
$${\frac{\partial^3}{\partial{x^3}}}
  {\frac{\partial^2}{\partial{y^2}}}
  ({{x^y}+{y^x}+{x^3}})$$
```

Output: [✓ TeX] XML verbose mode

Concatenate TeX [✓ Break lines] Page Width 12.0 Font Size 10

Position of operation sign: [✓ before line break] after line break duplicate [✓ indent after line break]

Figure 6: The MathML to TEX converter, stand-alone application interface

4.2 Algorithm overview

The subject of line-breaking for mathematical formulae is complex, and a full description of our approach is beyond the scope of this paper. For the purpose of this article, we highlight only some of the important aspects.

In a manner similar to the *total-fit* approach to line breaking implemented for TEX paragraphs by Knuth and Plass [21], our algorithm searches for all possible breakpoints in a formula and tries to find the combination of line breaks that will produce the best global arrangement.

In addition to splitting linear text, the method must take into account the two-dimensional nature of mathematical content and also consider the implicit semantics of expressions.

This leads to a number of constraints. For example, script sub-expressions may not be separated from their base expressions. Another common case is that of juxtaposition: Immediate function arguments should not be separated from the name of the function, *e.g.* we must avoid

$$\begin{array}{l} p(x-y,z)\ \sin \\ (2x+1)\ , \end{array}$$

but implicit multiplication can be split between the factors

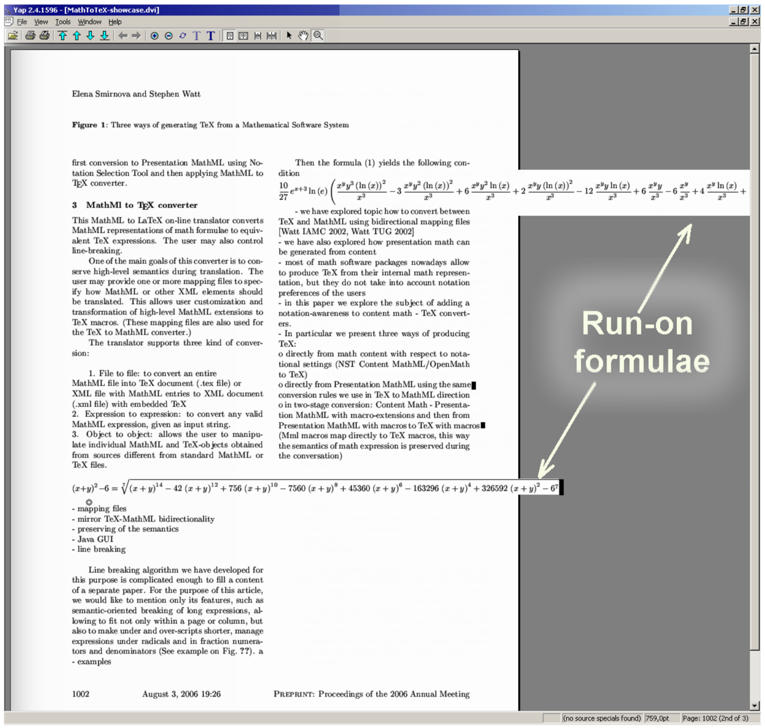
$$\begin{array}{l} p(x-y,z)\times \\ \sin(2x+1)\ . \end{array}$$

Juxtaposition is difficult to distinguish, since function application and implicit multiplication often have no explicit indication which operation is intended (even though MathML provides invisible operators for this purpose).

In general, every possible breakpoint is assigned a “penalty” value, so for example, signs and relations, such as $=$, $\Rightarrow$, *etc.*, as well as $+$ and $-$ are usually given priority over division and multiplication. Preference is given to breaking a formula closer to the root of the expression tree than within a branch. This means that a product of sums will preferentially break at the multiplications, unless the other penalties make this an overwhelmingly bad choice.

4.3 Customization

In the previous example we saw that when multiplication is expressed implicitly, line breaking may force the addition of an explicit multiplication sign. This may be represented by a central dot $\cdot$, times $\times$,


Yap 2.4.1596 - [MathToTeX-showcase.dvi]
File View Tools Window Help
Run-on formulae
(no source specials found) 759,0pt Page: 1002 (2nd of 3)

Elena Smirnova and Stephen Watt

Figure 1: Three ways of generating TeX from a Mathematical Software System

first conversion to Presentation MathML using Notation Selection Tool and then applying MathML to TeX converter.

3 MathMl to TeX converter

This MathML to LaTeX on-line translator converts MathML representations of math formulae to equivalent TeX expressions. The user may also control line-breaking.

One of the main goals of this converter is to conserve high-level semantics during translation. The user may provide one or more mapping files to specify how MathML or other XML elements should be translated. This allows user customization and transformation of high-level MathML extensions to TeX macros. (These mapping files are also used for the TeX to MathML converter.)

The translator supports three kind of conversion:

1. File to file: to convert an entire MathML file into TeX document (.tex file) or XML file with MathML entries to XML document (.xml file) with embedded TeX
2. Expression to expression: to convert any valid MathML expression, given as input string.
3. Object to object: allows the user to manipulate individual MathML and TeX-objects obtained from sources different from standard MathML or TeX files.

$$(x+y)^2-6=\sqrt[7]{(x+y)^{14}-42\,(x+y)^{12}+756\,(x+y)^{10}-7560\,(x+y)^{8}+45360\,(x+y)^{6}-163296\,(x+y)^{4}+326592\,(x+y)^{2}-6^7}$$

- mapping files
- mirror TeX-MathML bidirectionality
- preserving of the semantics
- Java GUI
- line breaking

Line breaking algorithm we have developed for this purpose is complicated enough to fill a content of a separate paper. For the purpose of this article, we would like to mention only its features, such as semantic-oriented breaking of long expressions, allowing to fit not only within a page or column, but also to make under and over-scripts shorter, manage expressions under radicals and in fraction numerators and denominators (See example on Fig. ??). a
- examples

Then the formula (1) yields the following condition

$$\frac{10}{27}e^{x+3}\ln(e)\left(\frac{x^y y^3(\ln(x))^2}{x^3}-3\,\frac{x^y y^2(\ln(x))^2}{x^3}+6\,\frac{x^y y^2\ln(x)}{x^3}+2\,\frac{x^y y(\ln(x))^2}{x^3}-12\,\frac{x^y y\ln(x)}{x^3}+6\,\frac{x^y y}{x^3}-6\,\frac{x^y}{x^3}+4\,\frac{x^y\ln(x)}{x^3}+\right.$$

- we have explored topic how to convert between TeX and MathML using bidirectional mapping files [Watt IAMC 2002, Watt TUG 2002]
- we have also explored how presentation math can be generated from content
- most of math software packages nowadays allow to produce TeX from their internal math representation, but they do not take into account notation preferences of the users
- in this paper we explore the subject of adding a notation-awareness to content math - TeX converters.
- In particular we present three ways of producing TeX:
o directly from math content with respect to notational settings (NST Content MathML/OpenMath to TeX)
o directly from Presentation MathML using the same conversion rules we use in TeX to MathML direction
o in two-stage conversion: Content Math - Presentation MathML with macro-extensions and then from Presentation MathML with macros to TeX with macros
(Mml macros map directly to TeX macros, this way the semantics of math expression is preserved during the conversation)

1002 August 3, 2006 19:26 PREPRINT: Proceedings of the 2006 Annual Meeting

Figure 7: TeX article without line breaking in long formulae.

asterisk $*$ or other operator $\otimes, \odot, \ldots$. Our package allows the user to specify which is preferred.

Likewise, the positioning of the operator at a line break can be set to the upper line, the lower line, or repeated on both lines (see Figure 8). In addition various indentation styles may be desired for the multiple lines. These options reflect different notational and cultural preferences, and can be customized by the user of the converter through the GUI interface or command line.

4.4 Line breaking in sub-expressions

As well as allowing a formula to fit within a given page or column width, the line breaking algorithm also handles situations where certain sub-expressions are too large for conventional line-breaking. These situations include long scripts (compare Figures 9 and 10), in fraction numerators and denominators (Figure 11), and expressions under radicals (Figure 12).

The sizes of the bounding boxes for the folded sub-expressions are normally calculated automatically, based on an optimal fit in the text area. If desired, they may instead be specified by the user. For example, for sub-scripts this is specified as a maximum ratio of the script width to the width of the base expression.

$$\begin{array}{l} x_1 - y_2 + z_3 + \\ \quad \alpha - \beta^2 + \gamma^3 + \\ \quad f(1) + g(2) \end{array}$$

(a) Before break

$$\begin{array}{l} x_1 - y_2 + z_3 \\ \quad +\alpha - \beta^2 + \gamma^3 \\ \quad +f(1) + g(2) \end{array}$$

(b) After break

$$\begin{array}{l} x_1 - y_2 + z_3 + \\ +\alpha - \beta^2 + \gamma^3 + \\ +f(1) + g(2) \end{array}$$

(c) Before and after break

Figure 8: Operator placement at line breaks

For large expressions, line breaking is a true two-dimensional problem involving box composition. In this setting, the height and width of individual terms depend on the choices made in displaying its sub-expressions. Examples of such nested line breaking are shown in Figures 10, 11 and 12.

As a practical implementation detail, we use the `array` environment to organize multi-line output in mathematical expressions. This is shown in

$$\sum_{(i,j)\in\{(\alpha,\beta)\ |\ ax^\alpha+bx^\beta=m,\ m\in R^+,\ \alpha\neq\beta,\ \gcd(a,b)=1\}} f(\ldots$$

Figure 9: Expression without line breaking does not fit in a text column

$$\sum_{(i,j)\in\left\{(\alpha,\beta)\ \middle|\ \begin{array}{l} ax^\alpha+bx^\beta=m, \\ m\in R^+,\ \alpha\neq\beta, \\ \gcd(a,b)=1 \end{array}\right\}} f(i)g^{-1}(j)\phi(i+j,i-j)$$

Figure 10: Adding line breaking in subscript allows the whole expression to fit in a column

$$\frac{\frac{10}{27}\,y^2\,\mathrm{e}^{(x+3)}\,\sqrt[3]{(x+3)}\,\mathrm{W}_0(1.5+2.5\,\imath)\times \left(\begin{array}{l} x^{(y-3)}\ln{(x)}^2\,y^3+6\,x^{(y-3)}\ln{(x)}\,y^2+ \\ 6\,x^{(y-3)}\,y-3\,x^{(y-3)}\ln{(x)}^2\,y^2- \\ 12\,x^{(y-3)}\ln{(x)}\,y-6\,x^{(y-3)}+ \\ 2\,x^{(y-3)}\ln{(x)}^2\,y+4\,x^{(y-3)}\ln{(x)}+ \\ y^{(x-2)}\,x^2\ln{(y)}^3-y^{(x-2)}\,x\ln{(y)}^3+ \\ 6\,y^{(x-2)}\,x\ln{(y)}^2+6\,y^{(x-2)}\ln{(y)}- \\ 3\,y^{(x-2)}\ln{(y)}^2-14\,y^{(x-2)}\,x\ln{(y)} \end{array}\right)}{(x+3)^3}$$

Figure 11: Line breaking in long fractions

$$(x+y)^2-6=\sqrt[7]{\begin{array}{l} (x+y)^{14}-42\ (x+y)^{12}+ \\ 756\ (x+y)^{10}-7560\ (x+y)^8+ \\ 45360\ (x+y)^6-163296\ \times \\ (x+y)^4+326592\ (x+y)^2-6^7 \end{array}}$$

Figure 12: Line breaking in long sub-radical expressions

```
$$
\begin{array}{l}
 \displaystyle A\,\frac{x+y}{3}+{14\,t}^{3}-\\
 \displaystyle B\,\frac{a-b}{4}+{12\,t}^{4}+\\
 \displaystyle C\,\frac{m+n}{5}+{10\,t}^{5}
\end{array}
$$
```

Figure 13: Encoding of multi-line constructions

$$\begin{array}{l} A\,\tfrac{x+y}{(a+b)^3}+14m^{12}- \\ B\,\tfrac{a-b}{(x-y)^3}+12n^{14} \end{array}$$

(a) In-line style

$$\begin{array}{l} \displaystyle A\,\frac{x+y}{(a+b)^3}+14m^{12}- \\ \displaystyle B\,\frac{a-b}{(x-y)^3}+12n^{14} \end{array}$$

(b) Display style

Figure 14: Array elements (a) without and (b) with explicit display style.

the code fragment of Figure 13. Note the command `\displaystyle` preceding every new line. This ensures rendering of fractions and scripts in display mode. To see the effect, compare the appearance of the formula with fractions of Figure 14a and Figure 14b.

4.5 Open questions

One of the challenges in line breaking is to distinguish implicit multiplication and application of unspecified or user-defined functions. We may assume function application in the cases of explicit markup or known functions, such as `\mathop{tg} \alpha` or `\tan x`. In general, however, we need either good heuristics, non-trivial semantic analysis, or explicit user markup.

Another challenge, this time from the aesthetic point of view, is how to arrange mixed expressions, such as shown in Figure 12, where we decide to maintain a single baseline for the overall expression and to align sub-expressions in place, instead of moving them to separate lines and splitting them there.

The final issue we mention is the question of page breaking in the case of multi-page content. This situation frequently arises with output of computer algebra systems. We have implemented an approach that takes page size into account, but there remain a large number of questions with respect to handling of subexpressions and layout choices.

5 Conclusions and future work

We have explored an alternative approach to generating TEX expressions from mathematical content when the content is presented in a conceptually oriented format.

The main idea of our approach is to maintain the mathematical markup at a high level, either in TEX or MathML, allowing extended markup for new mathematical concepts. This allows higher-level transformations among the formats and allows late binding of user-specified notational choices.

We have shown how the rendering of mathematical content with TEX can be customized with our Notation Selection Tool. For this, we considered two methods of conversion to TEX: One as a direct translation using features of the Notation Selection Tool. The second method was to use MathML extended with new elements. We showed that the second approach offers a more fine-grained control over the conversion process. It allows implicit semantics of mathematical expressions to be mapped from MathML template definitions to TEX macros. Additionally, it allows a line breaking implementation suitable for large, generated mathematical expressions.

We continue to explore certain open problems in the conversion between MathML and TEX. These include the automatic generation of templates for mapping rules and XSLT templates for notation conversions. We also wish to further investigate enhanced expression breaking methods in the presence of selectable notations. In particular, we intend to explore generating line-breaking hints for a late stage line breaker (*i.e.* one that operates after notation specialization). Another point of interest for our group in the MathML to TEX conversion area is in automated generation of TEX style files from XML cascading style sheets [22].

References

[1] *Maple User Manual*, Maplesoft, a division of Waterloo Maple Inc., 2005.

[2] *Mathematica*, Wolfram Research, Inc., 2004, `http://www.wolfram.com`.

[3] Richard D. Jenks and Robert Sutor, *AXIOM: the scientific computation system*, Springer-Verlag, New York, 1992.

[4] S.M. Watt, *Aldor*, pp. 265–270, in Handbook of Computer Algebra J. Grabmeier, E. Kaltofen, V. Weispfenning (editors) , Springer Verlag, Heidelberg, 2003.

[5] Mathematics on the Net, Symbolic Services, 2003, `http://www.orcca.on.ca/MONET/`.

[6] Mike Dewar, Elena Smirnova and Stephen M. Watt, *XML in Mathematical Web Services*, Proc. XML 2005 Conference — Syntax to Semantics, Nov 14–18, 2005, Atlanta GA, USA, `http://www.idealliance.org/proceedings/xml05/`.

[7] S. Buswell, O. Caprotti, D.P. Carlisle, M.C. Dewar, M. Gaetano, M. Kohlhase, et al., *The OpenMath Standard 2.0*, 2004, `http://www.openmath.org/cocoon/openmath/standard/om20/index.html`.

[8] R. Ausbrooks et 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`.

[9] Elena Smirnova and Stephen M. Watt, *Notation Selection in Mathematical Computing Environments*, pp. 339–355, Proc. Transgressive Computing 2006: A conference in honor of Jean Della Dora (TC 2006), April 24–26 2006, Granada, Spain.

[10] The notation selection on-line tool, 2002, `http://www.orcca.on.ca/MathML/NotationSelectionTool/`.

[11] William 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.

[12] MathML to TEX on-line converter, 2001, `http://www.orcca.on.ca/mathml/texmml/textomml.html`.

[13] Stephen M. Watt, *Exploiting Implicit Mathematical Semantics in Conversion between TEX and MathML*, Proc. Internet Accessible Mathematical Communication, (IAMC 2002), July 2002, Lille, France, `http://www.symbolicnet.org/conferences/iamc02`.

[14] S.M. Watt, *Conserving Implicit Mathematical Semantics in Conversion between TEX and MathML*, *TUGboat*, Vol. 23, No. 1, p. 108, 2002, `http://www.tug.org/TUGboat/Articles/tb23-1/watt.pdf`.

[15] E. Smirnova and S.M. Watt, *MathML to TEX Conversion: Conserving High-Level Semantics in Translation*, International Conference on MathML and Math on the Web (MathML 2002), June 28–30 2002, Chicago, USA, 2002, `http://www.mathmlconference.org/2002/presentations/smirnova/`.

[16] HTML to LATEX converter, `http://www.rpi.edu/~sofkam/html2latex/1.0/common/doc/html2latex.html`,

[17] TEX to MathML on-line converter, `http://www.orcca.on.ca/mathml/texmml/mmltotex.html`, 2001.

[18] 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.

[19] John Keith Foderaro, *Typesetting MACSYMA equations*, Proc. 2nd. MACSYMA User's Conference, June 1979.

[20] Michael Downes, *Breaking equations*, *TUGboat*, Volume 18, No. 3, Proceedings of the 1997 Annual Meeting, 1997, `http://www.tug.org/TUGboat/Articles/tb18-3/tb56down.pdf`.

[21] D.E. Knuth and M.F. Plass, *Breaking paragraphs into lines*, Software — Practice and Experience, 1981.

[22] *Cascading Style Sheets*, `www.w3.org/Style/CSS/`.