

Collecting Mathematical Expressions with Web Forms: Converting Plain Text to MathML with Plain2MathML

Clifford Johnston
West Chester University

Background

The software routines in Plain2MathML translate plain text or ASCII expressions into MathML. In particular, mathematical expressions collected through standard HTML forms can be converted into MathML and then shared with various applications including visual and audio rendering software. The Plain2MathML project is a compromise between the options of a graphical interface as in *WebEQ* and a comprehensive text interface like *TeX*. Since students have more experience typing mathematics using the syntax of graphing calculators or the somewhat less formalized "email math"¹ notation and ASCII documents are easily transmitted across the Internet, mathematics is often exchanged on the Internet in ASCII format. In his invited address to the NCTM, William Johnston noted that these common formats will likely continue to be used for the casual exchange of mathematics among students and instructors.² A plain text to MathML translator creates the possibility of displaying these exchanges with traditional notation as well as making them available for use in other applications.

In many cases, mathematical input accepted by forms is processed within the narrow parameters of the application requesting the input. Moreover, with application specific processing, the expression may be interpreted differently in different applications. Other applications do not process the mathematical input and leave the text as it is submitted. Discussion boards often use this method. While this is often satisfactory, it can be difficult to read complex expressions, and the format is often unintelligible by other applications. A set of common software functions for transforming plain text mathematics into MathML may provide some remedy to these problems and ease the exchange of mathematics on the web.³ While MathML is not currently supported in the popular browsers *Netscape* and *Microsoft Internet Explorer*, freely available programs such as *Techexplorer* and *WebEQ* offer a relatively simple methods to view MathML in both popular browsers. MathML also has the support of the leading vendors of mathematical software---Wolfram Research, Inc.; Waterloo Maple, Inc.; and Reduce. Since the Plain2MathML package is not a stand-alone application, developers can extend it to cover other areas as needed.

¹ As described on the [Math Forum Website](http://mathforum.org/dr.math/faq/faq.typing.math.html). "Ask Dr. Math: FAQ: Typing Math." [The Math Forum](http://mathforum.org/dr.math/faq/faq.typing.math.html). 1994. 7 Feb 2002. <<http://mathforum.org/dr.math/faq/faq.typing.math.html>>.

² Johnston, William I. "Principia Hypertextica: A Mathematics Educator's View of Web Design: Typesetting." 26 June 1997. Home page. 4 Apr. 2001 <<http://world.std.com/~wij/web-design/typesetting.html>>.

³ Bishop, Pam. "Report on MathML conference, University of Illinois, October 2000." [Learning and Teaching Support Network: Maths, Stats & OR Network](http://ltsn.mathstore.ac.uk/reports/mathml.htm#two). 5 Dec. 2000. U of Birmingham, U. K. <<http://ltsn.mathstore.ac.uk/reports/mathml.htm#two>>.

Goals

The design goals for the Plain2MathML project are:

1. To translate typical in-line ASCII mathematics notation as well as typical calculator syntax from graphing calculators, particularly the popular TI series;
2. To recognize expressions from elementary mathematics, algebra, and calculus;
3. To produce both presentation and content MathML;
4. To anticipate common "sloppy" notation often encountered in text representations of mathematics;
5. To function within applications and not as a stand-alone product; and
6. To adapt to other situations with minimal programming by the user.

In particular, adaptation should be accomplished through setting or resetting flags and not modifying internal routines. The Plain2MathML project is written as a PERL module. The module design allows a programmer easy access to the conversion routines. Moreover, the conversion process seems more suitable to server-side applications, for which PERL is a standard choice.

Function

The main conversion routines for Plain2MathML accept any ASCII string. For meaningful output, it should be either a mathematical expression or a body of text with mathematical expressions placed between double dollar signs ($\$$) as in *TeX*. The conversion routines can output either presentation or content MathML. The output can also include the appropriate coding for the applications *TechExplorer* or *WebEQ*. Finally, the output can be plain HTML. The output format is specified in the initial state or when the conversion routine is invoked. Finally, an XML namespace for the output can also be specified.

The Plain2MathML module, `plain2mml.pm`, is loaded into a PERL program in the usual way. The format of the primary routine call is

```
$mathml = plain2mml -> Plain2MathML($text,@options) .
```

Of the multiple difficulties in converting ASCII text to MathML, the following were of particular concern:

1. Differentiating symbols representing variables from symbols representing functions;
2. Interpreting "sloppy" notation;
3. Determining the scope of operations traditionally represented with infix notation such as + (plus), / (division), and ^ (power);
4. Encoding special operators from calculus, such as integration and summation, that may have ambiguous scope; and
5. Detecting invisible operations.

All of these problems result from ambiguity in the ASCII representation of mathematical expressions. The basic solution used in the Plain2MathML project was to make a choice between possible interpretations but allow the user of the module to select alternate interpretations by altering attributes of symbols and setting flags for the conversion process. In this way, someone using the module as part of an application can easily

override the conventions used by the module and make it conform to the conventions of their application. These conventions are set in an initial state that is created by the user before or at the time of a call to the conversion routine. Specific routines within the module allow the user to modify the initial state.

sions. For example, deciding if $a(x+1)$ should be interpreted as a times $x+1$ or a of $x+1$ depends on contextual clues of the surrounding text. Since it is considered likely that such expressions will be sent to the Plain2MathML routine without an contextual information, it is impossible for the conversion routine to decide among those interpretation. Related, but different, is the abuse of notation often used when writing mathematics. For example, neither Maple, Mathematica, nor the TI Calculators will accept the notation $\sin^2(x)$ for the usual interpretation as $(\sin(x))^2$. It was decided, however, that since this notation is so common in written mathematics, the module should be able to interpret that form of notation with its usual meaning. Under consideration is a method that would allow the user to set the level of "sloppiness" the module would attempt to handle.

The basic solution used in the Plain2MathML project was to make a choice between possible interpretations but allow the user of the module to select alternate interpretations by altering attributes of symbols and setting flags for the conversion process. In this way, someone using the module as part of an application can easily override the conventions used by the module and make it conform to the conventions of their application. These conventions are set in an initial state that is created by the user before or at the time of a call to the conversion routine. Specific routines within the module allow the user to modify the initial state.

Hence, as well as the main conversion routine, the module also exports the following routines:

```
$plain2mmlobj = plain2mml->new()  
$plain2mmlobj->Set(attribute,on/off)  
$plain2mmlobj->Add(type,token,properties)  
@tokenlist = $plain2mmlobj->List(type)  
@propertylist = $plain2mmlobj->ListProperties(token)
```

The module also includes standard POD documentation describing these routines.

Scope and interpretation of operators, particularly infix operators, was handled by making multiple passes of the expression looking for clues that would indicate the function and scope of the operator. Binding was predefined for various combinations of operators and then expanded during run time to handle interactions that are more complex. These binding rules handle operations whose scopes overlap or are adjacent. For example, $x^2/3$ is interpreted as $(x^2)/3$ rather than $x^{(2/3)}$. Separate routines handle the associative and the non-associative operations. Finally, certain tokens cause the conversion routine to abandon these binding rules in favor of rules for those particular operators. In particular, a class of functions flagged as "boundfunctions" interpret tokens such as \wedge and $_$ immediately following them as limits or parameters for the function.

Notation for integrals is particularly challenging, and a separate routine is used to preprocess the expression in integrals are present.

Determining invisible operations, such as multiplication in the expression $3x$, is relatively easy once the issues regarding ambiguity of tokens and the proper scope of operations are settled.

Example

The screen images, Figure 1 and Figure 2, show examples of an application using the Plain2MathML project.⁴ Figure 1 shows the expressions typed into an HTML form element. Figure 2 shows the output displayed with the *TechExplorer* plug-in. The presentation MathML produced for the first expression is:

```
<math xmlns='http://www.w3.org/1998/Math/MathML' mode='display'>
  <mrow>
    <mo>&Integral;</mo>
    <msup>
      <mi>x</mi>
      <mi>n</mi>
    </msup>
    <mo>&dd;</mo>
    <mrow>
      <mi>x</mi>
    </mrow>
    <mo>=</mo>
    <mfrac>
      <msup>
        <mi>x</mi>
        <mrow>
          <mi>n</mi>
          <mo>+</mo>
          <mn>1</mn>
        </mrow>
      </msup>
      <mrow>
        <mi>n</mi>
        <mo>+</mo>
        <mn>1</mn>
      </mrow>
    </mfrac>
    <mo>+</mo>
    <mi>C</mi>
  </mrow>
</math>
```

Applications

Two applications are currently using Plain2MathML. The WCU Mathematics Forum at West Chester University is a discussion board that uses Plain2MathML to allow students to preview mathematical expressions before posting as well as translating posted

⁴ This application is available at <http://math.wcupa.edu/~johnston/plain2mathml/MakeMathML.html>.

mathematics into MathML. The WebAssign project at North Carolina State University is using Plain2MathML as part of its homework delivery system.

Clifford Johnston
122 Anderson Hall, WCU
West Chester, PA 19383

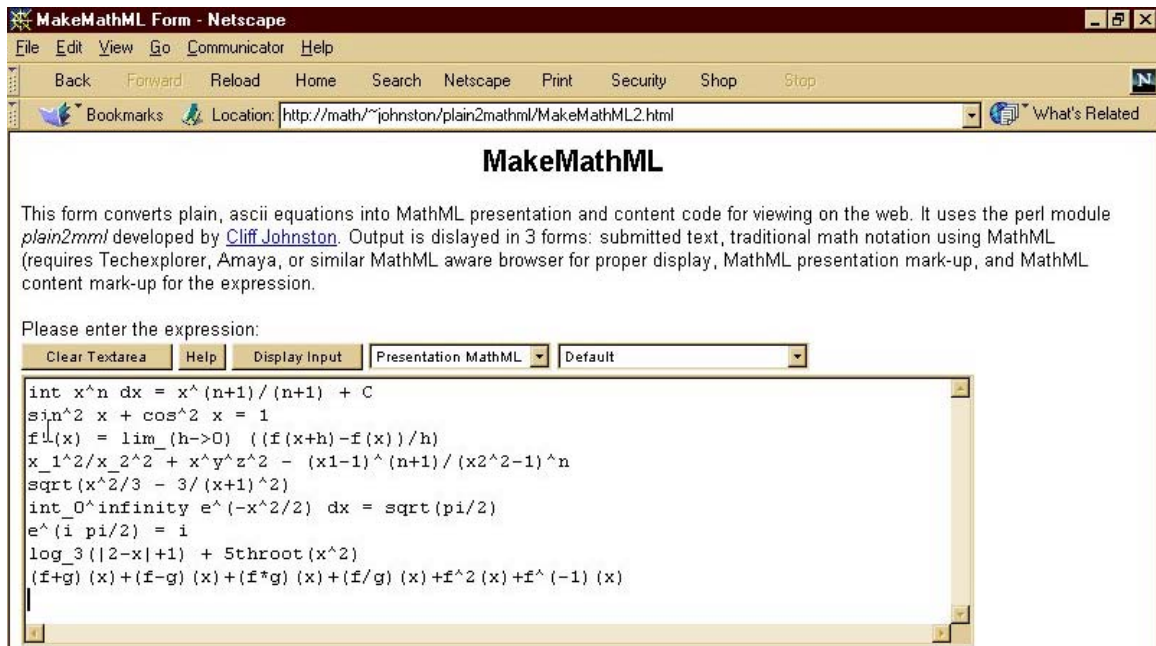


Figure 1

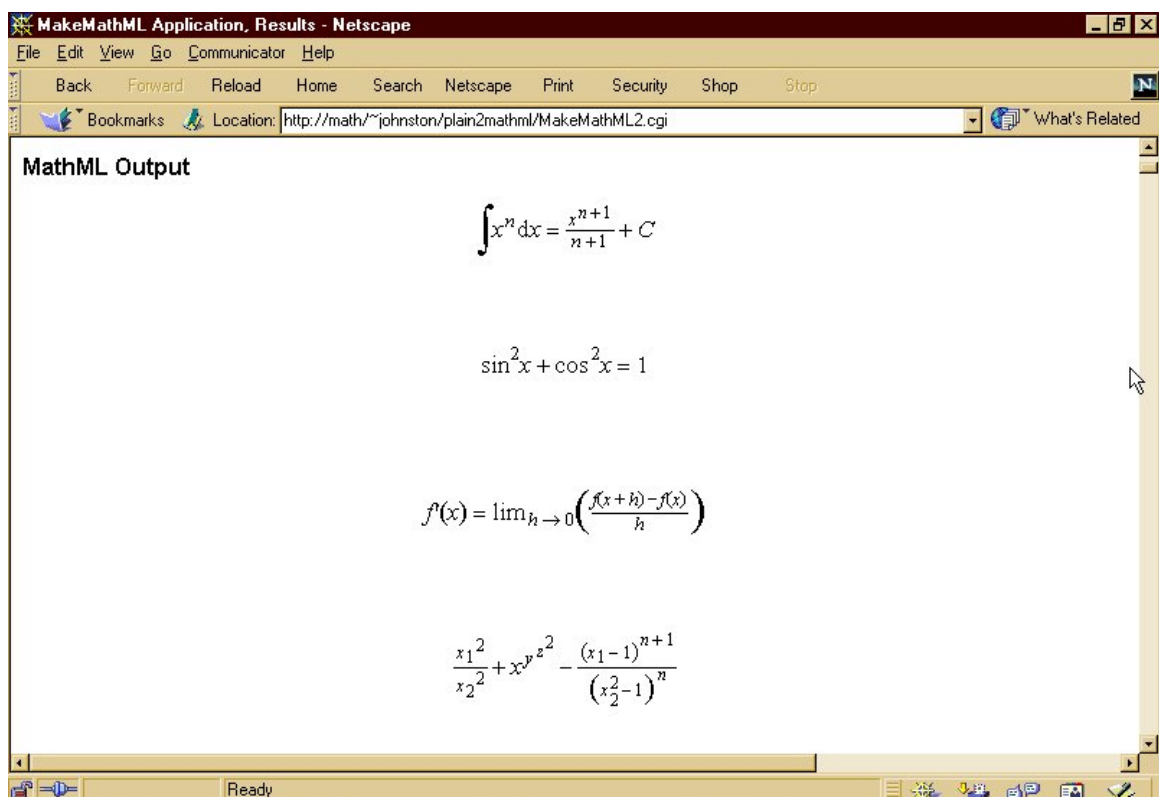


Figure 2