Strategies for Parallel Markup

Bruce R. Miller

Applied and Computational Mathematics Division
National Institute of Standards and Technology, Gaithersburg, MD, USA

Abstract. Cross-referenced parallel markup for mathematics allows the combination of both presentation and content representations while associating the components of each. Interesting applications are enabled by such arrangements: interaction with parts of the presentation to manipulate and query the corresponding content; enhanced search indexing. Although the idea of such markup is hardly new, effective techniques for creating and manipulating it are more difficult than it appears. Since the structures and tokens in the two formats often do not correspond one-to-one, decisions and heuristics must be developed to determine in which way each component refers to and is referred to by components of the other representation. Conversion between fine and coarse-grained parallel markup complicates XML identifier (ID) assignments. In this paper, we will describe the techniques developed for LaTeXML, a TeX/LaTeX to XML converter, to create cross-referenced parallel MathML. While not yet considering LaTeXML's content MathML to be truly useful, the current effort is a step towards that continuing goal.

1 Introduction

Parallel markup for mathematics provides the capability of providing alternative representations of the mathematical expression, in particular, both the presentation form of the mathematics, i.e. its appearance, along with the content form, i.e. its meaning or semantics. Cross-linking between the two forms provides the connection between them such that one can determine the meaning associated with every visible fragment of the presentation and, conversely, the visible manifestation of each semantic sub-expression. Thus cross-linked parallel markup provides not only the benefits of the presentation and content forms, individually, but support many other applications such as: hybrid search where both the presentation and content can be taken into account simultaneously; interactive applications where the visual representation forms part of the user-interface, but supports computations based on the content representation.

Of course, the *idea* of parallel markup is hardly new. The m:semantics element has been part of the MathML specification [1] since the first version, in 1998! What seems to be missing are effective strategies for creating, manipulating and using this markup. Fine-grained parallelism is when the smallest sub-expressions are represented in multiple forms, whereas with coarse-grained parallelism the entire expression appears in several forms. Fine-grained parallelism is generally

easier to create initially, and particularly when one deals with complex 'transfix' notations, or wants to preserve the appearance, but can infer the semantic intent of each sub-expression. Coarse-grained is often required by applications which may understand only a single format, or are unable to disentangle the fined-grained structure. HTML5 [3] only just barely accepts coarse-grained parallel markup, for example, by ignoring all but the first branch. Conversion from fine to coarse-grained is not inherently difficult, it can be carried out by a suitable walk of the expression tree for each format. But what isn't so clear is how to maintain the associations between the symbols and structures in the two trees. Indeed, there is typically no one-to-one correspondence between the elements of each format. Fine-grained parallelism, by itself, doesn't guarantee a clear association between all the symbols between the branches.

Our context here is LaTeXML, a converter from TeX/LaTeX to XML, and thence to web appropriate formats such as HTML, MathML and OpenMath. Input documents range from highly semantic markup such as sTeX [4], to intermediate such as used in the Digital Library of Mathematical Functions (DLMF) [6], to fairly undisciplined, purely presentational, markup as found on arXiv [8]. TeX induces high expectations for quality formatting forcing us to preserve the presentation of math. Meanwhile, the promise of global digital mathematics libraries and the potential reuse of a legacy of mathematics material encourages us to push as far as possible the extraction of content from such documents. At the very least, we should preserve whatever semantics is available in order to enable other technologies and research, such as LLaMaPuN [2], to resolve the remaining ambiguities. Getting ahead of ourselves, our first 'mini' strategy is just that: create fine-grained parallel markup at the first opportunity that presentation and semantics are available, in order to preserve them both.

In this paper, we describe the markup used in LaTeXML both for macros with known semantics, and for the result of parsing, and strategies for conversion to cross-linked, parallel markup combining Presentation MathML (pMML) and Content MathML (cMML). It should be noted that this does not mean that LaTeXML is producing useful quality cMML; the current work is a stepping stone towards that long-term goal.

Even in situations where only presentation markup is used, such as (currently) DLMF[1] where most symbols have always been hyper-linked to their definitions and the presentation-based search indexing is enhanced by the corresponding semantic content [7], the proposed strategies create a proper association between presentation and content resulting in a much cleaner implementation than the ad-hoc methods previously employed. Moreover, it is more complete, allowing the linkage to definitions to be extended to less textual operators such as binomials, floor, 3-j symbols, etc.

Listing 1.1. Internal representation of $a + F(a, b)$, after parsing to XMath (assuming F as a function)

```
<XMApp>
  <XMTok meaning="plus" role="ADDOP">+</XMTok>
  <XMTok role="ID" font="italic">a</XMTok>
  <XMDual>
    <XMApp>
      <XMRef idref="m1.1" />
      <XMRef idref="m1.2" />
      <XMRef idref="m1.3" />
    </XMApp>
    <XMApp>
      <XMTok role="FUNCTION" xml:id="m1.1" font="italic">F</XMTok>
      <XMWrap>
        <XMTok role="OPEN" stretchy="false">(</XMTok>
        <XMTok role="ID" xml:id="m1.2" font="italic">a</XMTok>
        <XMTok role="PUNCT">,</XMTok>
        <XMTok role="ID" xml:id="m1.3" font="italic">b</XMTok>
        <XMTok role="CLOSE" stretchy="false">)</XMTok>
      </XMWrap>
    </XMApp>
  </XMDual>
</XMApp>
```

2 Motivation

Before diving into examples, a brief introduction to LaTeXML's internal mathematics markup, informally called XMath, is in order. This markup, inspired by OpenMath and both pMML and cMML, is intentionally hybrid in order to capture both the presentation and content properties of the mathematical objects throughout the step-wise processing from raw TeX markup, through parsing and, ultimately, semantic annotation. The main elements of concern are:

XMApp generalized application (think m:apply or om:OMA);
XMTok generalized token (think m:mi, m:mo, m:mn, m:csymbol);
XMDual parallel markup container of the content and presentation branches;
XMRef shares nodes between branches of XMDual, via xml:id and idref attributes;
XMWrap container of unparsed sequences of tokens or subtrees (think m:mrow).

(Please see the online manual[2] for more details.)

By way of motivation, consider the simple example in Listing 1.1. The role attribute on tokens indicates the syntactic role that it plays in the grammar; in this case, we've asserted that F is a function, allowing the expression to be parsed. At the top-level, the sum requires no special parallel treatment since the presentation for infix operators is trivially derived from the content form (i.e. the application of '+' to its arguments). The application of F to its arguments benefits somewhat from parallel markup. This is a typical situation with the fine-grained XMDual: the content branch is the application of some function or

[1] http://dlmf.nist.gov/
[2] http://dlmf.nist.gov/LaTeXML/manual/

Listing 1.2. MathML representation of $a + F(a, b)$

```
<math display="block" alttext="a+F(a,b)" class="ltx_Math" id="m1">
  <semantics id="m1a">
    <mrow xref="m1.7.cmml" id="m1.7">
      <mi xref="m1.4.cmml" id="m1.4">a</mi>
      <mo xref="m1.5.cmml" id="m1.5">+</mo>
      <mrow xref="m1.6.cmml" id="m1.6d">
        <mi xref="m1.1.cmml" id="m1.1">F</mi>
        <mo xref="m1.6.cmml" id="m1.6e">&ApplyFunction;</mo>
        <mrow xref="m1.6.cmml" id="m1.6c">
          <mo xref="m1.6.cmml" id="m1.6" stretchy="false">(</mo>
          <mi xref="m1.2.cmml" id="m1.2">a</mi>
          <mo xref="m1.6.cmml" id="m1.6a">,</mo>
          <mi xref="m1.3.cmml" id="m1.3">b</mi>
          <mo xref="m1.6.cmml" id="m1.6b" stretchy="false">)</mo>
        </mrow>
      </mrow>
    </mrow>
    <annotation-xml id="m1b" encoding="MathML-Content">
      <apply xref="m1.7" id="m1.7.cmml">
        <plus xref="m1.5" id="m1.5.cmml"/>
        <ci xref="m1.4" id="m1.4.cmml">a</ci>
        <apply xref="m1.6d" id="m1.6.cmml">
          <ci xref="m1.1" id="m1.1.cmml">F</ci>
          <ci xref="m1.2" id="m1.2.cmml">a</ci>
          <ci xref="m1.3" id="m1.3.cmml">b</ci>
        </apply>
      </apply>
    </annotation-xml>
    <annotation id="m1c" encoding="application/x-tex">a+F(a,b)</annotation>
  </semantics>
</math>
```

operator (here F) to arguments (here a, b), but they are represented indirectly using XMRef to point to the corresponding sub-expressions within the presentation. While one could represent the delimiters and punctuation as attributes (as in MathML's m:mfenced), that loses attributes of those attributes such as stretchiness, size or even color. A more compelling case is made by complex transfix notations or semantic macros, as we will shortly see.

However, this simple example already hints at a hidden complexity. Converting to either pMML and cMML is straightforward (given rules for mapping XMath elements to MathML): simply walk the tree, depth-first, following each XMRef to the referenced node and choosing the first or second branch of XMDual for content or presentation, respectively. Even cross-linking is straightforward in the absence of XMDual, when the generated content or presentation nodes are 'sourced' from the same XMath node (F, a, and b, in the example): simply assign ID's to the source XMath node and the generated nodes; record the association between them; afterwards, the presentation and content nodes that were sourced from the same ID are connected by getting an xref attribute referring each to the other. But with XMDual one has not only to determine when the generated nodes are related, one has to contend with extra tokens; in the example, the parentheses and comma appear only in the presentation. Presumably, those tokens

should be associated with the *application* of F, as would the containing m:mrow. The desired result is shown in Listing 1.2.

A fuller illustration of the issues encountered in typical LaTeX markup combines complex transfix notations and semantic macros, such as:

```
\left\langle\Psi\middle|\mathcal{H}\middle|\Phi\right\rangle
 +  \defint{a}{b}{F(x)}{x}
```

This example, whose internal form is shown in Listing 1.3, involves quantum-mechanics notations, which LaTeXML's parser is happily able to recognize. Additionally, we've introduced a semantic macro `\defint` to represent definite integration, which will be transformed to so-called 'Pragmatic' Content MathML form, to enhance the illustration with a many-to-many correspondence. (The implementation of `\defint` is not difficult, but outside the scope of this article)

3 Main Strategies

We will see that a key part of the method is determining which nodes are visible to presentation, to content or to both branches. This is easily determined by an algorithm like mark-and-sweep garbage collection. Simply traverse the tree from the root following the first branch of each XMDual to mark all nodes as visible to presentation, and repeat for the second branch to mark content visibility. The analogy is apt, as this also allows pruning of nodes that are not visible at all, as sometimes occurs with complex macro usage — another mini-strategy.

A few definitions will also be relevant. During the depth-first tree traversal transforming XMath into either pMML or cMML, the *current node* is the XMath node being transformed. The *current container* is the XMDual node (if any) containing the current node; in the context of this discussion, an XMath container is always the application of some function or operator. We'll call the latter the *current operator*.

We will ascribe to each generated target node (pMML or cMML) an XMath *source node* that can be considered 'responsible' for the target node. In the common, simplest case, a current node that is visible to both branches and generates a token node in the target can be used as the source node.

A special case occurs when the target MathML element is a container; these generally do not correspond to symbols, and ought to be associated with the nearest application (think m:apply or m:mrow)[3]. In this case, the source should be the nearest ancestor XMDual of the current node, that is the *current container*.

Similar reasoning applies when a token (non-container) symbol is generated from an XMath token which is not visible to the opposite branch; typically the notational icing of some transfix. We presume that is the only visible manifestation of the *current operator*, and so that is taken as the source node.

[3] Exceptions are m:msqrt or m:menclose where they tend to represent *both* the application of an operation and yet are the only visible manifestation of the operator! However, we also note that a common use of cross-linking in HTML is to turn them into href links; but HTML does not allow nested links!

Listing 1.3. Internal representation of $\langle\Psi|\mathcal{H}|\Phi\rangle + \int_a^b F(x)dx$ as XMath

```
<XMApp>
  <XMTok meaning="plus" role="ADDOP">+</XMTok>
  <XMDual>
    <XMApp>
      <XMTok meaning="quantum-operator-product" />
      <XMRef idref="m2.5" />
      <XMRef idref="m2.6" />
      <XMRef idref="m2.7" />
    </XMApp>
    <XMWrap>
      <XMTok role="OPEN">⟨</XMTok>
      <XMTok role="ID" xml:id="m2.5">Ψ</XMTok>
      <XMTok role="CLOSE" stretchy="true">|</XMTok>
      <XMTok role="ID" xml:id="m2.6" font="caligraphic">H</XMTok>
      <XMTok role="OPEN" stretchy="true">|</XMTok>
      <XMTok role="ID" xml:id="m2.7">Φ</XMTok>
      <XMTok role="CLOSE">⟩</XMTok>
    </XMWrap>
  </XMDual>
  <XMDual>
    <XMApp>
      <XMTok meaning="hack-definite-integral" role="UNKNOWN" />
      <XMRef idref="m2.1" />
      <XMRef idref="m2.2" />
      <XMRef idref="m2.3" />
      <XMRef idref="m2.4" />
    </XMApp>
    <XMApp>
      <XMApp>
        <XMTok role="SUPERSCRIPTOP" scriptpos="post2" />
        <XMApp>
          <XMTok role="SUBSCRIPTOP" scriptpos="post2" />
          <XMTok mathstyle="display" meaning="integral" role="INTOP">∫</XMTok>
          <XMTok role="ID" xml:id="m2.1" font="italic">a</XMTok>
        </XMApp>
        <XMTok role="ID" xml:id="m2.2" font="italic">b</XMTok>
      </XMApp>
      <XMApp>
        <XMTok meaning="times" role="MULOP">⁢</XMTok>
        <XMDual xml:id="m2.3">
          <XMApp>
            <XMRef idref="m2.3.1" />
            <XMRef idref="m2.3.2" />
          </XMApp>
          <XMApp>
            <XMTok role="FUNCTION" xml:id="m2.3.1" font="italic">F</XMTok>
            <XMWrap>
              <XMTok role="OPEN" stretchy="false">(</XMTok>
              <XMTok role="UNKNOWN" xml:id="m2.3.2" font="italic">x</XMTok>
              <XMTok role="CLOSE" stretchy="false">)</XMTok>
            </XMWrap>
          </XMApp>
        </XMDual>
        <XMApp>
          <XMTok meaning="differential-d" role="DIFFOP" font="italic">d</XMTok>
          <XMTok role="UNKNOWN" xml:id="m2.4" font="italic">x</XMTok>
        </XMApp>
      </XMApp>
    </XMApp>
  </XMDual>
</XMApp>
```

In the example, the angle brackets and vertical bars are thus ascribed to the `quantum-operator-product` operator.

In pseudo-code, the *source* node for a given *target* is thus:

```
if target is a container
  if current container exists
    current container
  else
    current
else if target is visible in both branches
  current
else if current container exists
  if current operator is hidden from presentation
    current operator
  else
    current container
else
  current
```

Once this ascription of source nodes is done, the cross-referencing between the generated targets is easily established: the xref of a pMML (cMML) node is the cMML (pMML, respectively) node that was ascribed to the same source XMath node; if multiple nodes were ascribed to that source node, the first target node, in document order, is the sensible choice. Applying this method to the example from Listing 1.3 yields Listing 1.4, where we can see, for example, that the angle brackets and vertical bars are associated with the `quantum-operator-product` operator while the various m:bvar, m:lowlimit, etc, are properly associated with the integral, *not* the integral operator.

4 Outlook

We have described a set of strategies for generating parallel markup with cross-references consisting of encouraging fine-grained parallel structures, mark-and-sweep to detect nodes visible to presentation or content and to garbage-collect, and a method to determine related nodes within each branch of the parallel markup.

Parallel markup must also be adapted to larger structures such as eqnarray, and AMS alignments with intertext containing multiple formula and/or document-level text markup. While the fundamental issue is the same — separating presentation and content forms — this seems to demand a distributed markup that separates the presentation and content forms into distinct math containers. LaTeXML currently has an ad-hoc, but not entirely satisfactory solution for this, but we will experiment with adapting the methods described here. However, it remains to be seen whether cross-referencing across separate math containers can be made useful.

And, now that generating Content MathML is more fun, we must continue working towards generating *good* Content MathML. Ongoing work will attempt

Listing 1.4. MathML representation of $\langle\Psi|\mathcal{H}|\Phi\rangle + \int_a^b F(x)dx$

```
<math display="block" alttext="..." class="ltx_Math" id="m2">
  <semantics id="m2a">
    <mrow xref="m2.13.cmml" id="m2.13">
      <mrow xref="m2.9.cmml" id="m2.9">
        <mo xref="m2.8.cmml" id="m2.8">&LeftAngleBracket;</mo>
        <mi mathvariant="normal" xref="m2.5.cmml" id="m2.5">&Psi;</mi>
        <mo xref="m2.8.cmml" id="m2.8a" stretchy="true" fence="true">|</mo>
        <mi xref="m2.6.cmml" id="m2.6" class="ltx_font_mathcaligraphic">&HilbertSpace;</mi>
        <mo xref="m2.8.cmml" id="m2.8b" stretchy="true" fence="true">|</mo>
        <mi mathvariant="normal" xref="m2.7.cmml" id="m2.7">&Phi;</mi>
        <mo xref="m2.8.cmml" id="m2.8c">&RightAngleBracket;</mo>
      </mrow>
      <mo xref="m2.10.cmml" id="m2.10">+</mo>
      <mrow xref="m2.12.cmml" id="m2.12c">
        <msubsup xref="m2.12.cmml" id="m2.12">
          <mo xref="m2.11.cmml" id="m2.11" symmetric="true" largeop="true">&int;</mo>
          <mi xref="m2.1.cmml" id="m2.1">a</mi>
          <mi xref="m2.2.cmml" id="m2.2">b</mi>
        </msubsup>
        <mrow xref="m2.12.cmml" id="m2.12b">
          <mrow xref="m2.3.cmml" id="m2.3c">
            <mi xref="m2.3.1.cmml" id="m2.3.1">F</mi>
            <mo xref="m2.3.cmml" id="m2.3d">&ApplyFunction;</mo>
            <mrow xref="m2.3.cmml" id="m2.3b">
              <mo xref="m2.3.cmml" id="m2.3" stretchy="false">(</mo>
              <mi xref="m2.3.2.cmml" id="m2.3.2">x</mi>
              <mo xref="m2.3.cmml" id="m2.3a" stretchy="false">)</mo>
            </mrow>
          </mrow>
          <mo xref="m2.11.cmml" id="m2.11a">&InvisibleTimes;</mo>
          <mrow xref="m2.12.cmml" id="m2.12a">
            <mo xref="m2.11.cmml" id="m2.11b">d</mo>
            <mi xref="m2.4.cmml" id="m2.4">x</mi>
          </mrow>
        </mrow>
      </mrow>
    </mrow>
    <annotation-xml id="m2b" encoding="MathML-Content">
      <apply xref="m2.13" id="m2.13.cmml">
        <plus xref="m2.10" id="m2.10.cmml"/>
        <apply xref="m2.9" id="m2.9.cmml">
          <csymbol xref="m2.8" id="m2.8.cmml" cd="latexml">quantum-operator-product</csymbol>
          <ci xref="m2.5" id="m2.5.cmml">normal-&Psi;</ci>
          <ci xref="m2.6" id="m2.6.cmml">&HilbertSpace;</ci>
          <ci xref="m2.7" id="m2.7.cmml">normal-&Phi;</ci>
        </apply>
        <apply xref="m2.12c" id="m2.12.cmml">
          <int xref="m2.11" id="m2.11.cmml"/>
          <bvar xref="m2.12c" id="m2.12a.cmml">
            <ci xref="m2.4" id="m2.4.cmml">x</ci>
          </bvar>
          <lowlimit xref="m2.12c" id="m2.12b.cmml">
            <ci xref="m2.1" id="m2.1.cmml">a</ci>
          </lowlimit>
          <uplimit xref="m2.12c" id="m2.12c.cmml">
            <ci xref="m2.2" id="m2.2.cmml">b</ci>
          </uplimit>
          <apply xref="m2.3c" id="m2.3.cmml">
            <ci xref="m2.3.1" id="m2.3.1.cmml">F</ci>
            <ci xref="m2.3.2" id="m2.3.2.cmml">x</ci>
          </apply>
        </apply>
      </apply>
    </annotation-xml>
    <annotation id="m2c" encoding="application/x-tex">...</annotation>
  </semantics>
</math>
```

to establish appropriate OpenMath Content Dictionaries, probably in a Flexi-Formal sense [5], improved math grammar, and exploring semantic analysis.

References

1. Ausbrooks, R., Buswell, S., Carlisle, D., Chavchanidze, G., Dalmas, S., Devitt, S., Diaz, A., Dooley, S., Hunter, R., Ion, P., Kohlhase, M., Lazrek, A., Libbrecht, P., Miller, B., Miner, R., Sargent, M., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical Markup Language (MathML) version 3.0. W3C Recommendation, World Wide Web Consortium (W3C) (2010), `http://www.w3.org/TR/MathML3`
2. Ginev, D., Jucovschi, C., Anca, S., Grigore, M., David, C., Kohlhase, M.: An architecture for linguistic and semantic analysis on the arXMLiv corpus. In: Applications of Semantic Technologies (AST) Workshop at Informatik 2009 (2009).
3. Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E.D., O'Connor, E., Pfeiffer, S.: HTML5. W3C Recommentation, World Wide Web Consortium (W3C) (2014), `http://www.w3.org/TR/html5/`
4. Kohlhase, M.: Using LaTeX as a semantic markup format. Mathematics in Computer Science 2(2), 279–304 (2008).
5. Kohlhase, M.: The flexiformalist manifesto. In: Voronkov, A., Negru, V., Ida, T., Jebelean, T., Petcu, D., ane Daniela Zaharie, S.M.W. (eds.) 14th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012). pp. 30–36. IEEE Press, Timisoara, Romania (2013).
6. Miller, B.R., Youssef, A.: Technical aspects of the digital library of mathematical functions. Annals of Mathematics and Artificial Intelligence 38(1-3), 121–136 (2003).
7. Miller, B.R., Youssef, A.: Augmenting presentation mathml for search. In: Proceedings of Conference on Intelligent Computer Mathematics. Lecture Notes in Artificial Intelligence, vol. 5144. Springer, Heidelberg (2008).
8. Stamerjohanns, H., Kohlhase, M., Ginev, D., David, C., Miller, B.: Transforming large collections of scientific publications to XML. Mathematics in Computer Science 3(3), 299–307 (2010).