Towards ARIA Standards for Mathematical Markup

MathJax Consortium *
Davide Cervone[1], Peter Krautzberger[2], and VolkerSorge[3]

[1] Union College, NY, dpvc@union.edu
[2] krautzource UG, peter.krautzberger@mathjax.org
[3] University of Birmingham, UK, V.Sorge@cs.bham.ac.uk
www.mathjax.org

Abstract

While MathML is a successful standard for the representation of mathematical syntax and semantics, its lack of browser support and failure to synchronise with advancing web technologies makes its future as a web standard doubtful. Consequently there has been a need for alternative rendering solutions, that transform mathematics into web content actually supported by browsers. To make this content accessible assistive technology must work with mathematics regardless of the actual implementation of the visual rendering. We present work in MathJax that semantically enriches Presentation MathML and exposes the enrichment into rendering elements thereby allowing an accessibility tool to work across the whole palette of rendering options provided by MathJax, essentially independent of MathJax itself. We also present some ideas how this work could be leveraged towards establishing a refined ARIA standard for Mathematics. The paper therefore aims to stimulate discussion on how Mathematics could be made readily available for assistive technology in modern web design.

1 Introduction

Although Mathematical formulas on the web can be represented in their own specialised markup language, MathML [2] as part of the HTML5 standard [1], only very few major browsers implement MathML rendering natively, leaving the support for displaying formulas included in pure MathML on web pages sketchy. Those browsers that provide MathML support, generally only offer it partially or lack full integration into other technologies of the Open Web Platform. For example, Firefox's MathML implementation does not implement the GlobalEventHandler API and inconsistent support for CSS styles on MathML elements. While Safari's Webkit engine has a more modern integration into standard DOM (the tree structure representing the HTML document) interfaces, it does not support a number of MathML elements, such as maction, or collapses line-broken content onto the same line. This leaves mathematical expressions as artefacts of the DOM that are immune to modern web technology standards. Also, neither browser is supporting the MathML constructs for elementary mathematics, such as long divisions. The situation becomes even more dire when considering eBook readers that should implement the epub3 standard which requires MathML support; only very few offer any support for mathematics (cf. [3]). As neither Mozilla nor Apple are actively developing the MathML support in their browsers, any progress, including bug fixes, is left to unreliable volunteer efforts. Finally, Google Chrome and Microsoft IE/Edge (or 55%-75% of the browser market) list MathML support as simply not planned.

Therefore, while MathML is a valuable XML markup language, useful for both the syntactic composition of expressions via Presentation MathML and full semantic representation via Content MathML, its future as a web standard is doubtful. This is compounded by the fact it is in most parts made obsolete by advances in the HTML5 standard and the possibilities provided by CSS that can simulate most, if not all of MathML functionality. As a consequence authors of mathematical web content and platforms supporting mathematical authoring rely on JavaScript libraries such as MathJax [6] to ensure flawless rendering of formulas across all platforms and browsers. However, this leads to a difficult situation for making this content fully accessible as it disconnects visual and assistive technology (AT) rendering completely. While MathJax can provide high-quality cross-browser visual rendering, it has to decouple accessible rendering by providing raw MathML visually hidden for third party AT systems. Alternatively it can implement its own assistive extension that can concentrate exclusively on Maths accessibility.

*This work was partially supported by the Alfred P. Sloan Foundation.

For the former MathJax users still require MathML-aware AT. This is non-trivial as MathML in the wild consists almost exclusively of Presentation MathML which is not accessible on its own. To make it accessible, AT solutions need sophisticated heuristics to generate semantic information about the mathematical layout. In practice, most tools outsource the problem to MathPlayer [10] but Apple VoiceOver and Google ChromeVox both provide such features as well. This continues to pose a significant hurdle preventing other AT solutions to enter the market and thus holds back widespread and competitive AT support for Mathematics.

For the latter MathJax is developing a complete accessibility solution that works in all browsers providing improved reflow, navigation, collapse and expansion, selective highlighting and dynamic speech text generation. Since the support has to work across the entire palette of rendering solutions MathJax provides (including SVG support, HTML5 and CSS, native MathML, etc.) we need a common way of generating this information, storing it in the DOM and exposing it selectively and effectively independent of MathJax. We do this by semantically enriching Presentation MathML and, by extension, also LaTeX and AsciiMath, and embedding it in the DOM elements via data attributes that provide a fast and standardised means of retrieving information from the DOM, fully consistent with HTML5 practices. However, at the moment this support can only be described as ad hoc as it depends on internal interpretations and representations of mathematics rather than on well-defined open standards, such as the Accessible Rich Internet Applications standard (WAI-ARIA or ARIA for short) [7]. ARIA allows the semantic annotation of web components so their meaning and behaviour can be adequately conveyed to people with disabilities by assistive technologies. Unfortunately, ARIA only specifies a global `math` role for web content and is therefore of extremely limited use in our context. However, we believe that our developments can inform emerging web standards development regarding improvements to the way Mathematics can be handled. Rather than embedding semantic information into HTML via a bespoke separate markup language like Content MathML, we advocate the development of a hierarchy of ARIA elements that can be applied to arbitrary HTML constructs and that convey sufficient mathematical meaning to enable direct adoption from AT technologies of the future.

2 Semantic Enrichment

MathJax's aim of improving web accessibility of mathematics with a consistent user experience across all its rendering solutions rely on a semantic enrichment of Presentation MathML elements. It is based on a heuristic rewriting of the syntactic structure of MathML elements into a semantic tree, that aims to stay faithful to the given notation, without fixing too much of the semantics to avoid false interpretations. While this leads to a more shallow interpretation than a full blown semantic markup language like Content MathML [2], it has the advantage that it retains effectively all the components of the original expression. While in Content MathML symbols like parentheses are omitted and entire expressions are replaced by their semantic counterpart, we aim to retain these as they are important for visual and aural rendering.

The basic problem we face can already be demonstrated with the example of binomial coefficients: $\binom{n}{k}$ It can be modelled in Presentation MathML in a number of different ways, of which three are presented in Figure 1. The middle one, which abuses the fraction notation, is indeed the one recommended by the standard [2][1].

It is obvious that in order to have this element read out correctly as a binomial coefficient by a screen reader one has to employ some heuristic interpretation. And this is indeed the role of the semantic tree interpretation we employ. While it was originally implemented for the ChromeVox screen reader [11], we now embed it directly into the Presentation MathML via data attributes. This has two advantages: Firstly, it allows us to expose the semantic interpretation with limited, conservative remodelling of the original MathML expression, using a fast and standardised means of retrieving information from the DOM. Secondly, the attributes can be preserved when rendering the maths expression in browsers that do not natively support MathML.

Figure 2 presents the enriched MathML expression. The attributes have been shortened by omitting the "`data-semantic-`" prefix to preserve space. They represent effectively the structure of the semantic tree, via the parent, children and content attributes, as well as types and roles our heuristics have inferred.

[1] `http://www.w3.org/TR/MathML3/chapter3.html#id.3.3.2.2`

```
<math>                 <math>                         <math>
  <mfenced>              <mfenced>                      <mfenced>
    <mstack>               <mfrac linethickness="0pt">    <mtable>
      <mi>n</mi>             <mi>n</mi>                     <mtr><mtd><mi>n</mi></mtd></mtr>
      <mi>k</mi>             <mi>k</mi>                     <mtr><mtd><mi>k</mi></mtd></mtr>
    </mstack>              </mfrac>                       </mtable>
  </mfenced>             </mfenced>                     </mfenced>
</math>                </math>                        </math>
```

Figure 1: Three different MathML representations for binomial coefficients.

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow type="vector" role="binomial" id="6" children="2,5" content="7,8" parent="9"
       speech="StartBinomialOrMatrix n Choose k EndBinomialOrMatrix">
    <mo type="fence" role="open" id="7" parent="6" speech="left-parenthesis">(</mo>
    <mtable>
      <mtr type="line" role="binomial" id="2" children="0" parent="6"><mtd>
          <mi type="identifier" role="latinletter" id="0" parent="1" speech="n">n</mi>
      </mtd></mtr>
      <mtr type="line" role="binomial" id="5" children="3" parent="6"><mtd>
          <mi type="identifier" role="latinletter" id="3" parent="4" speech="k">k</mi>
      </mtd></mtr>
    </mtable>
    <mo type="fence" role="close" id="8" parent="6" speech="right-parenthesis">)</mo>
  </mrow>
</math>
```

Figure 2: Enriched MathML expression for binomial coefficient.

Types can be understood as an immutable property of an expression, while roles are mutable, in that they are assigned and can be changed during the semantic interpretation. For example an identifier can simply have role Latin letter or, if more information is available, for example get the role function assigned. For more details on the semantic tree and enrichment procedure see [4].

Note, that the speech attribute is not part of the semantic tree, but is a pre-computed speech string, which can also be computed on the fly from the embedded semantic information. This is part of the novel accessibility solution that we have developed for MathJax. It exploits the embedded structure to offer direct aural rendering of the entire expression via the computed speech string, but also interactive exploration of the sub-expressions, via the recursively embedded speech information. In addition, it allows for selective highlighting and contrast change as well as collapsing of large sub-expressions and their selective expansion. For a more detailed introduction of all the accessibility features see [5].

Since MathJax provides rendering in all browsers, and most will not render native MathML, we need a means to provide the same accessibility experience for all our rendering solutions, without having to use the enriched MathML as a parallel structure. Data attributes can achieve this, as we can exploit them in all HTML5 elements. Figures 3 and 4 show parts of the DOM elements produced by MathJax's CommonHTML and SVG rendering engine, respectively. The CommonHTML renderer uses `span`s styled with CSS to provide identical markup and layout across browsers and platforms. Note, that the representations have been abbreviated and some attributes have been omitted to preserve space.

The main point, however, is that the information embedded in the DOM structure, whether it is MathML, SVG or pure HTML, provides a uniform means for both aural rendering and navigating the structure by assistive technology. Hence the embedded semantics plays a role which is very similar to the way ARIA provides information for assistive technology components, regardless of how a particular visual component is implemented. To further motivate the need for an orthogonal structure to indicate mathematical meaning in DOM elements we discuss some of the problems on implementing MathML in a modern web context.

3 MathML in a web context

After the rejection of the `<math>` element in HTML3.2, MathML was formed as a separate XML spec. As such, it was immensely successful outside the web, providing a versatile and powerful solution for XML work-flows. This success and the lack of browser implementations led to a disconnect from other web technologies, most importantly CSS, which provides layout in browser engines, and WAI ARIA, which

```
<span aria-hidden="true" role="math" class="mjx-math" id="MJXc-Node-48">
  <span class="mjx-mrow" id="MJXc-Node-49">
    <span speech="StartBinomialOrMatrix n Choose k EndBinomialOrMatrix"
          parent="9" content="7,8" children="2,5" id="6" role="binomial" type="vector">
      <span speech="left-parenthesis" parent="6" id="7" role="open" type="fence">
        <span style="padding-top: 1.225em; padding-bottom: 1.224em;">(</span>
      </span>
      <span style="vertical-align: -0.971em; padding: 0px 0.167em;">
        <span class="mjx-table">
          <span speech="n" parent="6" children="0" id="2" role="binomial" type="line">
            <span style="padding: 0px 0px 0.221em; width: 0.6em;">
              <span style="margin-top: 0.333em; margin-bottom: 0.164em;">
                <span speech="n" parent="1" id="0" role="latinletter" type="identifier">
                  <span style="padding-top: 0.217em; padding-bottom: 0.286em;">n</span>
                </span>
    ...
</span>
```

Figure 3: Abbreviated DOM structure produced by MathJax's CommonHTML renderer.

```
<svg aria-hidden="true" focusable="false" role="img">
  <g transform="matrix(1 0 0 -1 0 0)" stroke-width="0" stroke="currentColor">
    <g speech="StartBinomialOrMatrix n Choose k EndBinomialOrMatrix"
       parent="9" content="7,8" children="2,5" id="6" role="binomial" type="vector">
      <use speech="left-parenthesis" parent="6" id="7" role="open" type="fence"
           xlink:href ="#MJSZ3-28"></use>
      <g transform="translate(903,0)">
        <g transform="translate(-18,0)">
          <g transform="translate(0,665)">
            <g speech="n" parent="1" id="0" role="latinletter" type="identifier">
              <use xlink:href="#MJMATHI-6E"></use>
            </g>
          </g>
    ...
</svg>
```

Figure 4: Abbreviated SVG output produced by MathJax's SVG renderer.

provides semantic information for accessibility purposes.

The relationship between MathML and CSS is a difficult one. For example, [9] argued that styling via CSS should be limited, e.g., to the root, and styling of mathematics be exclusively realised via MathML's internal styling methodology. Miner gives two arguments: a practical argument (implementations do not support CSS styling) and a theoretical argument (styling can influence semantics, e.g., bold or Gothic characters carry meaning). Reversely, [12] argued that layout engines need to limit the CSS that is permissible for MathML nodes, even going as far as calling math layout "fundamentally an unsolvable problem". We disagree with both sides. For widespread use and consistent programming interfaces, math layout needs to be realised as just another piece of CSS layout. This cannot lead to limiting the CSS cascade in any form.

Miner's first argument is fortunately out of date; while not trivial, it is possible today to fully support styling due to modern Web APIs such as `getComputedStyle`. Miner's second argument is more philosophical. We disagree with it on two grounds. First, the problem of CSS changing output unpredictably is nothing particular to mathematics. A regular paragraph would become just as confusing to the reader if it was accidentally styled like a heading or a footnote. The idea of conveying meaning via styling is a flawed approach in mathematics just as it is in any other subject domain. Styling can (and should) aid the visual user but cannot substitute some form of semantic markup. Second, the web is a new medium and is developing unique stylistic customs; mathematics should benefit from the web's expanded expressiveness rather than artificially limit itself to the customs of print layout. For example, bolding a mathematical expression alongside other content (e.g., in a heading) is common on the web as can be seen from the usage of MathJax's autobold extension. It seems, in this particular medium, authors prefer the lack of stylistic disruption to the risk of causing confusion; this is most likely due to the fact that such confusion is simply not an issue on the web. Similarly, where traditional layout is limited to various typefaces, authors can be much more expressive on the web.

From discussions with developers on both Gecko and WebKit, Vafai's argument seem far too general. The discussion that led to his article took place on the Chromium bug tracker but is unfortunately not publicly available due to security concerns. The discussion centred exclusively around the problem of stretchy fences. While this is clearly an unusual problem in a CSS context, it seems questionable to describe it as a fundamental problem, especially since Gecko implements stretchy characters in a CSS context and has done so for years without any issues. But even if it was a fundamental problem, then this should lead to a requirements analysis for math layout in a CSS context rather than a blind rejection. A suitable comparison might be the changes to CSS to enable bidirectional and vertical text layout which also led to fundamental changes to the internal concept of direction. Similarly, MathML might have to be modified to allow implementation in today's CSS engines.

In general, it is clear from both poly-filling MathML as well as the state of existing implementations that high quality math layout can be realised in CSS engines.

A significant issue from the historic disconnect between MathML and CSS are incompatibilities. For example, `<mpadded>` does not behave like CSS `padding`, `<mtable>` does not support all border types of `<table>` and has subtle differences in handling relative column sizes. This makes it difficult to develop consistently for the web and makes implementations overly complex.

Furthermore, mathematical layout can only be achieved using MathML elements now. However, the intended layout for HTML elements (e.g., table, paragraph, heading) has been abstracted into CSS which allows authors to leverage individual layout features without being forced into using the specifics of any particular element. This has allowed new design paradigms to develop. For example, modern responsive grid layout shares many properties with tables but is not limited by the use of `<table>` elements and has in turn given rise to new concepts such as the CSS Flexible Box Layout module and the CSS Grid Layout module. Math layout, on the other hand, continues to be restricted to MathML elements where implemented. This limits the potential for mathematics for the web as it is far less flexible than its HTML/CSS counterparts.

4 Towards WAI-ARIA for Maths: A Basis for Discussion

Where CSS abstracts and extends the traditional layout of HTML elements, the WAI-ARIA specification abstracts and extends the semantics. The ARIA model provides a pragmatic orthogonal method for exposing semantics in situations where the HTML elements are not usable. Typical examples include buttons and forms where design and usability restrictions can make the correct HTML element problematic to use while the semantic information remains crucial for human and machine readers.

While MathML splits into Presentation and Content MathML, this split does not match the situation of HTML. On the one hand, Presentation MathML is has several semantic elements (e.g., `<msqrt>`, `<mfrac>`, `<mlongdiv>`) and reliable heuristics exist (e.g., in AT) to extract more detailed semantic information. On the other hand, Content MathML, while suitable for information exchange among computational systems, makes it difficult to capture the semantics necessary for human readers; it is also rarely used in the wild).

Therefore, similar to the problems caused by the lack of CSS modules for math layout, the restriction of mathematical semantics to Content MathML elements severely limits the usability of mathematics on the web. First, Content MathML does not lend itself to styling for presentation but has to be transformed extensively to provide a useful visual representation. This is fundamentally different from HTML elements where CSS modules exist to realise the visual rendering directly. Second, limiting semantics to the use of elements limits their potential.

We believe MathML is sorely missing this orthogonal method. As browser implementations are limited and mostly unavailable, the lack of an ARIA module for mathematics adds an enormous burden for authors and developers who have to either lock their users into very specific system requirements (e.g., IE with MathPlayer) or generate accessible representations in a separate step which is often very expensive. Alternatives, are often only ad hoc solutions. For example, Gecko has implemented a few non-standard roles for mathematics, [8] via XML roles. On the web, MathML rendering solutions like MathJax have to implement complex tools for exposing semantic information of the layout to AT if they want to overcome the disconnect of visual and AT rendering and provide basic accessibility features such as exploration.

We believe an initial set of ARIA roles to subsume Presentation MathML would not have to be very large. By avoiding encoding general mathematical knowledge, it can focus on the structure of presentational layout and Presentation MathML notation. Given the overlap with HTML, many MathML elements should be mapped to their natural counterparts (or corresponding ARIA roles), e.g., `mtable` with `table`, `mglpyh` with `img`; elements and attributes for styling (e.g., `mstyle, mspace, mpadded, malign`) should be mapped to CSS. In addition, existing ARIA roles are available, e.g., the ARIA `role=error` subsumes `merror`, `role=application` subsumes `maction`, `role=separator` subsumes fraction lines, `msline`, and can improve MathML notation (decimal separators, set notation); similarly, `role=grid` subsumes the "stacking" elements (`munderover`, `mstack`, `mfrac` etc.). Finally, a few elements are redundant, e.g., grouping elements (`mrow`,`msrow`, `msgroup`), shorthands (`msup`, `msup` etc.,) textual elements (`mtext`, `ms`). This leads to a very concise first set of candidate for ARIA roles, in addition to the existing `role=math`:

- `fraction` for fractions and fraction-like content; subsumes `mfrac`)
- `enclosed-by`; subsumes `menclose`, `mroot, msqrt`, `mfenced` (and other representations of fences such as the equivalent `mi+mo`).
- `identifier`, `operator`, `number`; subsumes `mi`, `mo`, `mn`.
- `multiscripts` (pre-/-post sub/sup)
- `separator` (exists) for `mfrac` and `mstack` lines.
- `dividend` and `divisor`, `minuend` and `subtrahend`, `summand`, `carry` for elementary math.

A comparison with Presentation MathML shows that this list would cover the gaps left by de-duplication and existing aria roles as described earlier. This short list, together with de-duplication and existing ARIA roles as described, would provide complete basic support for all elements and attributes of Presentation MathML. The goal is then to extend this "rump" system to a minimal skeleton of roles that would be efficient to implement. By this we mean they should allow assistive technology to get access to the structure of mathematical layout in HTML or SVG and it should remove the need for complex heuristics. Such a skeleton might start by adding roles such as: (a) `relation`, (b) `pre/in/post`-fix operator, (c) `operand` for the operands of an `operator`, (d) `numerator` and `denominator` as sub-roles to `fraction`, (e) `lim`, `lim-begin`, `lim-end` for limit notation, (f) `matrix` and `vector` due to their importance.

The idea of a properly developed ARIA module for mathematics has additional potential. There are examples of HTML elements (e.g., main, nav, s, i) that have been (re-)introduced to HTML because their corresponding ARIA roles proved critical in real world content. Similarly, adding a set of ARIA role for mathematics might not only help the situation of math in the browser but also provide two directions for improvement: to improve MathML as the ARIA module can be more easily extended from real world use cases as well as to improve HTML by providing a path for elements into HTML which helps build a stable, long term future for math markup into browser engines.

References

[1] Hypertext Markup Language (html) version 5.0. W3c candidate recommendation, World Wide Web Consortium, 2013. Available at `http://www.w3.org/TR/html5`.

[2] D. Carlisle, P. Ion, R. Miner. MathML 3.0. W3c recommendation, 2010. `http://www.w3.org/TR/MathML3`.

[3] Diagram Center. Product matrices. `diagramcenter.org/research/product-matrices-complete.html`.

[4] D. Cervone, P. Krautzberger, and V. Sorge. Towards meaningful visual abstraction of mathematical notation. In *Proceedings of 10th Workshop on Mathematical User Interfaces*, 2015.

[5] D. Cervone, P. Krautzberger, and V. Sorge. Employing semantic analysis for enhanced accessibility features in mathjax. In *Accessible Devices and Services*, 2016. Submitted.

[6] MathJax Consortium. MathJax version 2.5, 2014. `http://www.mathjax.org`.

[7] J. Craig and M. Cooper. WAI-ARIA 1.0. W3c recommendation, 2014. `http://www.w3.org/TR/wai-aria`.

[8] J. Diggs. Rfc: Use of xml-roles object attribute for mathml without aria. W3C Mailing List Post, 2015. `https://lists.w3.org/Archives/Public/public-pfwg/2015Jul/0051.html`.

[9] R. Miner. Re: Problem with mathplayer and css. W3C Mailing List Post, 2003. `http://lists.w3.org/Archives/Public/www-math/2003Jan/0012.html`.

[10] N. Soiffer. Mathplayer: web-based math accessibility. *Conf. on Computers and accessibility*. ACM, 2005.

[11] V. Sorge, C. Chen, T.V. Raman, and D. Tseng. Towards making mathematics a first class citizen in general screen readers. In *11th Web for All Conference*, Seoul, Korea, 6–9 April 2014. ACM.

[12] Ojan Vafai. Mathml in webkit. Blog Post, 2013. `http://web.archive.org/web/20150224234431/http://ojanvafai.com/post/mathml-in-webkit`.