Employing Semantic Analysis for Enhanced Accessibility Features in MathJax

Davide Cervone
MathJax Consortium
Union College, NY
Email: dpvc@union.edu

Peter Krautzberger
MathJax Consortium
krautzource UG
Email: peter.krautzberger@mathjax.org

Volker Sorge
MathJax Consortium
University of Birmingham, UK
Email: V.Sorge@cs.bham.ac.uk

URL: www.mathjax.org

***Abstract*—Recent changes in the landscape for assistive technology solutions for Mathematics on the web have prompted the development of MathJax into a single rendering and accessibility solution. We present our current efforts that depend on a novel semantic interpretation of Presentation MathML expressions. This allows us to introduce a new notion of responsive equations, on which we build advanced accessibility features with improved reflow of content, selective highlighting, and dynamic speech text generation, as well as an innovative interaction technique based on abstracting and intelligently summarising sub-expressions.**

***Index Terms*—STEM Accessibility, Mathematics, MathJax**

I. Introduction

The text-to-speech translation of mathematical expressions has always been a challenging problem and is one major obstacle for fully inclusive education. Consequently, a number of software solutions have been researched over the years (see [1] for an overview). As web delivery grows, web-accessibility for mathematics is more important than ever.

Although formulas can be represented in their own specialised markup language (MathML [3], part of HTML5 [4]), only Firefox and Safari partially support it. Besides low market shares [19] they fall short in functionality such as basic elements (e.g., `maction` on Safari) or APIs (e.g., GlobalEventHandler on Firefox). Thus support for displaying formulas represented in MathML on web pages is sketchy. Rendering tools like MathJax [5] can provide high-quality cross-browser rendering but have to decouple accessible rendering from visual rendering by providing raw MathML visually hidden.

To overcome this problem, 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 Presentation MathML is not expressive enough to provide accessible rendering, we developed heuristics for semantically enriching Presentation MathML and, by extension, any other MathJax-compatible input such as LaTeX and AsciiMath. These heuristics use MathML's tree structure to build a tightly related semantic tree that is injected into the presentation elements. This allows us to easily expose the identified semantic structure in the DOM, and this structure is used in turn to generate alternate representations such as speech text, and when combined with the underlying Presentation MathML tree, responsive rendering and exploration interfaces.

II. Math accessibility on the Web

Before 2012, math accessibility on the web was synonymous with MathPlayer [2] on Internet Explorer. With IE11 deprecating plugins like MathPlayer, and ChromeVox [6] and VoiceOver [7] adding some MathML support, the landscape grew more complex. Unfortunately, the core problem for web-based math accessibility has not changed: browsers offer very limited MathML support. Consequently mathematics on the web comes in the following flavors: 1) **Pure MathML markup** relying on the user to view the page with one of the few MathML-capable browsers. 2) **Pre-rendered binary images** to ensure correct, low-quality display, at most embedding the original markup in alt-text. 3) **Rendered with MathJax** so the web page author ensures mathematical content is rendered independent of the browser. 4) In a more recent trend, **pre-rendered markup** (usually with MathJax running on NodeJS) ensures correct display by including HTML (with CSS) or SVG in the web page.

Hence rendering solutions like MathJax remain critical for authors and users, becoming the de facto standard for Mathematics rendering on the web in the past five years.

```
<math>
  <mi>a</mi>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
  <mo>+</mo>
  <mi>b</mi>
  <mi>x</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo>=</mo>
  <mn>0</mn>
</math>
```

Fig. 1. MathML representation of the quadratic equation.

```
<span class="math" id="MathJax-Span-7" role="math" style="width: 8.246em; display: inline-block;">
  <span style="display: inline-block; position: relative; width: 7.115em; height: 0px; font-size: 116%;">
    <span style="position: absolute; clip: rect(1.457em 1000em 2.751em -999.997em); top: -2.53em; left: 0.003em;">
      <span class="mrow" id="MathJax-Span-8">
        <span class="mi" id="MathJax-Span-9" style="font-family: STIXGeneral; font-style: italic;">a</span>
        <span class="msubsup" id="MathJax-Span-10">
          <span style="display: inline-block; position: relative; width: 0.919em; height: 0px;">
            <span style="position: absolute; clip: rect(3.397em 1000em 4.151em -999.997em); top: -3.984em; left: 0.003em;">
              <span class="mi" id="MathJax-Span-11" style="font-family: STIXGeneral; font-style: italic;">x
                <span style="display: inline-block; overflow: hidden; height: 1px; width: 0.003em;"></span>
              </span>
              <span style="display: inline-block; width: 0px; height: 3.99em;"></span>
            </span>
            <span style="position: absolute; top: -4.415em; left: 0.488em;">
              <span class="mn" id="MathJax-Span-12" style="font-size: 70.7%; font-family: STIXGeneral;">2</span>
              <span style="display: inline-block; width: 0px; height: 3.99em;"></span>
            </span>
          </span>
        </span>

   ...
</span>
```

Fig. 2. Abbreviated MathJax HTML representation of the quadratic formula.

MathJax covers 85% of the MathML3 test suite and provides input processors for commonly used languages such as TEX/LATEX and AsciiMath. It is used by most major scientific and educational publishers as well as large platforms such as StackExchange, edX, Quora, and Wikipedia. MathJax's free CDN service sees over 4.5 million unique daily visitors.

It is difficult for MathJax to provide fully accessible output, however, as the only web standard for math accessibility is indeed MathML. ARIA, for example, does not provide roles for exposing mathematical information beyond the global math role. But since MathML can not be rendered natively in many browsers, MathJax has to carefully separate accessible MathML from the visual output. To illustrate this point, consider the simple example of the quadratic equation $ax^2 + bx + c = 0$. Its MathML expression is given in Fig. 1. While this can be exposed to assistive technology (AT), e.g., to VoiceOver or ChromeVox directly and to other AT vendors via the new MathPlayer library, it must be hidden from the browsers themselves, as they usually cannot render it. On the other hand, MathJax's actual visual rendering (given in HTML/CSS format in Fig. 2), must be hidden from the AT to avoid it producing gibberish.

Since most mathematics on the web is rendered by MathJax, this provides a unique opportunity to combine accessibility features with its visual rendering. MathJax already provides basic AT features such as magnification and global scaling (see section IV). Both can negatively affect users with learning disabilities, in particular those who benefit from a less overloaded layout. Highlighting suffers from similar limitations as one can at best highlight the underlying MathML tree, which may not necessarily group sub-expressions semantically.

The speech translation of mathematics poses unique difficulties that make it a more challenging task than the aural rendering of text. Difficulties include: the wide spectrum of symbols, use and importance of punctuation as well as white space, and complex two-dimensional math layout. While there exist rule sets for translating mathematical expressions into speech (e.g., MathSpeak [8]) they usually require some interpretation of the mathematics that goes beyond what Presentation MathML alone can offer. We therefore have designed a novel semantic enrichment method for MathML that allows the implementation of advanced accessibility features in MathJax.

III. Semantic Enrichment

Our main improvements for accessible rendering of mathematics rely on a semantic enrichment of presentation MathML elements. This is based on a heuristic analysis of the syntactic structure of MathML elements as a semantic tree, and aims to stay faithful to the given notation while avoiding false semantic interpretations as much as possible. Although this leads to a more shallow interpretation than a full-blown semantic markup language like Content MathML [3], 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 counterparts, we aim to retain these as they are important for both visual and aural rendering. Moreover, it allows us to combine the semantic interpretation with the presentation form by embedding it via HTML attributes, with limited, conservative remodelling of the original MathML expression. This allows the presentation element tree to provide a different view of the MathML expression without having to maintain a separate structure in parallel, which would be necessary if exploiting MathML's `semantics` tag.

While there exist methods for translating mathematical syntax into Content MathML, they either restrict the mathematics to what a particular computer algebra system can handle [9], [10] or are based solely on LATEX interpretation [11], [12], [13]. Our semantic tree, on the other hand, is built to handle any mathematical notation found in arbitrary web documents. We first briefly sketch the major ideas of the semantic-tree generation — an extension of the heuristics implemented in

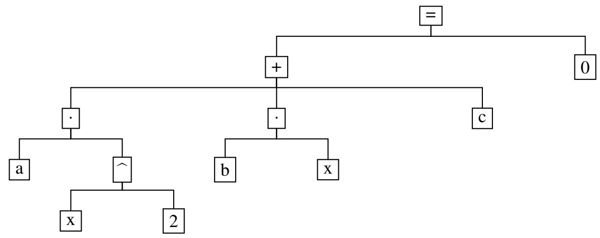

Fig. 3. Semantic term tree for the quadratic equation.

the screen reader ChromeVox [6] — and then describe how its information is integrated into existing MathML structures.

A. Semantic Tree

The main problem for semantic enrichment of Presentation MathML is to transform its flat structure into one that correctly determines the scope of operators, relations, etc. Our approach aims to represent a formula in a semantic tree structure akin to a term tree. The semantic tree is assembled bottom-up, where we first classify the single components of an expression, giving each an immutable type and a mutable role. The former aims to capture the basic nature of the symbol, while the latter is used to describe the role of a symbol in the context of the formula. For example, f, which has the type of "identifier", gets a default role of "Latin letter" assigned while no additional information is known. Once more knowledge of its semantic meaning is available, its role is refined. For example, in the expression $f(x)$ it would get the role of a "function", while its role remains unchanged in $f + g$.

A central heuristic then builds term trees from flat structures by promoting relations and defining operator precedence orders as well as determining properly delimited structures. As an example of this heuristic we observe how the quadratic equation $ax^2 + bx + c = 0$ is rewritten from Presentation MathML (Fig. 1) into its semantic interpretation (Fig. 3).

Observe that the transformation tries hard to recognise elided multiplications. In addition, our procedure contains a number of heuristics, in particular to (1) determine potential function applications, (2) break up symbol sequences into elided products, (3) recognise scope and nesting of big operators (e.g., sums, integrals), (4) distinguish tables into matrices, vectors, and case statements, (5) combine punctuated expressions and determine the meaning of ellipses.

Technically the tree is constructed by analysing MathML elements, interpreting their type, role, and font, and turning them into semantic nodes with parent pointer and a variable number of children. In addition, we have a notion of content elements for each node. This is a possibly empty list of semantic nodes that are combined or abstracted over by this particular node. For example, a node representing the application of a single operator like $+$ to a variable number of summands will have only the semantic nodes representing the summands as children, while retaining all the intermediate occurrences of $+$ in its list of content nodes. This allows us to keep a connection between nodes of the semantic tree and elements in the original MathML structure for tasks like synchronised highlighting or the semantic enrichment we will discuss in the next section.

B. Embedding into MathML

The basic idea of embedding the semantic tree into MathML is by modelling the components of the tree via additional data attributes in the individual elements of the MathML expression. Data attributes provide a fast and standardised means of retrieving information from the DOM (the tree structure representing the HTML document), which is fully consistent with HTML5 practices.

We add new data attributes to reflect both content and structure of the semantic tree. The former are attributes reflecting type, role, and font information stored in each node of the tree. The latter effectively provide each node with a unique semantic id, and, if necessary, a parent pointer and lists of pointers to children and content nodes. In addition, we have attributes that provide administrative information with respect to artefacts that have been introduced or omitted due to the mapping onto the MathML expression.

In the majority of cases the embedding is straight forward, with as little modification to the original MathML expression as possible; however, this can not always be maintained for more complex structures and the original expression must be augmented or partially rewritten. This often requires refining the tree structure by adding extra `mrow` tags for grouping, adding invisible elements representing omitted operations, or providing additional attributes with administrative information on parts of the semantic tree that were collapsed or expanded when mapped onto the original MathML.

Fig. 4 presents the enriched MathML for the quadratic equation. The added attributes represent the structure and information of the semantic tree. Note that all attributes are prefixed by `data-semantic-`, which has been omitted for easier reading and to preserve space. Observe that the MathML expression now contains both extra groupings and invisible-times applications, with the latter marked as newly added.

IV. Responsive Mathematical Content

We leverage the embedded semantic tree to create *responsive equations* that help us to improve reflow of expressions for better support of small screens and magnification. In general, responsive design focuses on optimising content for different display sizes, not only by re-arranging content, but also by transforming content itself, such as cropping images [14], abstracting icons [15], or modifying tables [16]. Reflowing mathematics poses a great challenge as it combines the properties of text, tables, and graphics into a single problem. While good line-breaking algorithms exist for print, they are often counter-productive on the web, damaging legibility of larger equations beyond repair. The problem is exacerbated by the fact that content is created with print in mind, manually fitting it to page dimensions. Manual line breaks, arrangements across tabular layout, and other such *tweaks* make a sensible reflow harder to accomplish.

```
<math type="relseq" role="equality" id="16" children="15,10"
    content="9">
 <mrow type="infixop" role="addition" id="15" children="12,14,8"
     content="4,7" parent="16">
  <mrow type="infixop" role="implicit" id="12" children="0,3"
      content="11" parent="15">
   <mi type="identifier" role="latin" id="0" parent="12">a</mi>
   <mo type="operator" role="multiplication" id="11" parent="12"
     added="true">&#x2062;</mo>
   <msup type="superscript" role="latin" id="3" children="1,2"
       parent="12">
    <mi type="identifier" role="latin" id="1" parent="3">x</mi>
    <mn type="number" role="integer" id="2" parent="3">2</mn>
   </msup>
  </mrow>
  <mo type="operator" role="addition" id="4" parent="15">+</mo>
  <mrow type="infixop" role="implicit" id="14" children="5,6"
      content="13" parent="15">
   <mi type="identifier" role="latin" id="5" parent="14">b</mi>
   <mo type="operator" role="multiplication" id="13" parent="14"
     added="true">&#x2062;</mo>
   <mi type="identifier" role="latin" id="6" parent="14">x</mi>
  </mrow>
  <mo type="operator" role="addition" id="7" parent="15">+</mo>
  <mi type="identifier" role="latin" id="8" parent="15">c</mi>
 </mrow>
 <mo type="relation" role="equality" id="9" parent="16">=</mo>
 <mn type="number" role="integer" id="10" parent="16">0</mn>
</math>
```

Fig. 4. Semantically enriched MathML for the quadratic equation.

MathJax supports magnification in two ways: first, standard browser zoom is supported by re-rendering mathematics at the new size. Second, MathJax offers a "lens" that allows further magnification of individual mathematical expressions and provides global scaling settings to enlarge all math elements at once. On small devices, zooming usually is not beneficial, as it leads to overflow and two-dimensional scrolling. For zooming and magnification, it is possible to avoid excessive scrolling or panning by reflowing with line-breaks; but applying line-breaking programmatically to long or complex equations often destroys their readability due to cluttered results. We exploit the semantically enriched markup to perform line-breaks at mathematically appropriate positions only. While this can occasionally lead to less optimal usage of screen real-estate, it does significantly help readability of formulas and also yields good results on small form factors.

We demonstrate the effects with an equation found on math.stackexchange.com, which aligns with our focus on handling arbitrary content well, not on handling well-prepared content excellently. Fig. 5 shows the formula with regular line-breaking, while Fig. 6 has it with semantically informed line-breaking. Observe that the latter breaks regularly at equation symbols and does not split individual summands apart.

V. Semantic Highlighting

While magnification aims primarily to support readers with low vision, we can also exploit the semantic features to assist readers with particular reading difficulties like dyslexia. The choice of high-contrast colours [17] and selective highlighting [18] can be helpful for reading comprehension.

Although it is not difficult for MathJax to change colours to high contrast via CSS styles, this is not necessarily sufficient. Mathematical expressions are generally large collections of mostly unconnected symbols in two-dimensional layout and can therefore be particularly daunting for readers with dyslexia. Thus it can be beneficial to enable readers to get selective highlighting of sub-expressions of a complex formula, for example by hovering over parts of an expression. While in principle it is possible to use the basic DOM structure to identify sub-trees, reading comprehension is enhanced by highlighting mathematically meaningful sub-formulas instead.

To illustrate this, consider the first line from Fig. 6. Highlighting purely on the syntactic tree, we see that the first level of the MathML tree is effectively a single row of symbols or combined elements like fractions and sub- or superscripts. Below we use alternating background colours for the elements that could be highlighted individually.

$$I_\nu(\nu^{-1},1)=\frac{\pi^2}{4}\ln\left(\frac{(1+\nu)^{1+\nu}}{\nu^\nu}\right)-\frac{7\zeta(3)}{8}\nu+2\int_1^{\frac{1-\nu}{1+\nu}}\frac{\chi_3(v)}{(1+v)^2}\,\mathrm{d}v$$

Using semantic information, on the other hand, highlighting is done on the level of the relation and additive operations tree. Consequently we end up with a much smaller number of sub-expressions and clearer selection of highlighted sub-formulas.

$$I_\nu(\nu^{-1},1)=\frac{\pi^2}{4}\ln\left(\frac{(1+\nu)^{1+\nu}}{\nu^\nu}\right)-\frac{7\zeta(3)}{8}\nu+2\int_1^{\frac{1-\nu}{1+\nu}}\frac{\chi_3(v)}{(1+v)^2}\,\mathrm{d}v$$

In practice, sub-expressions are highlighted when hovering over them with the mouse pointer, or by switching highlighting on permanently. It also can be recursively refined for sub-expressions, e.g., hovering on the denominator or numerator of a fraction only. For permanent highlighting, the opaqueness of the background gradually increases in nested sub-expressions. Nevertheless, in large expressions, highlighting only helps to some degree to get an overview of the structural composition of a formula; to aid this further, we introduce a technique for structural abstraction in the next section.

VI. Structural Abstraction

Users with reading challenges, such as dylexia, can quickly become overhelmed by the complex two-dimensional layout of mathematical expressions. Such readers can be assisted by simplifying the structure of the formula initially and letting them individually explore the equation by manually expanding selected sub-expressions.

We have implemented this via a user interface using MathML's `maction` element (cf. [3, 3.7.1]). It allows users to explore the content using click, keyboard, and touch events. `maction` elements are nested so that only the next level of the collapse is revealed. The element indicating collapsed content is a simple Unicode construction, ◀X▶, with X indicating the top-level structure that was collapsed. For example we use ◀+▶ to indicate a sum, ◀$\int$▶ for an integral, etc.

Fig. 7 demonstrates four different states of collapse of our example equation. To determine which parts are collapsed, our algorithm somewhat surprisingly does not calculate sub-expression width since these are not available before rendering.

$$
\begin{aligned}
I_\nu(\nu^{-1},1) &= \underbrace{\frac{\pi^2}{4}\ln\left(\frac{(1+\nu)^{1+\nu}}{\nu^\nu}\right) - \frac{7\zeta(3)}{8}\nu}_{\text{Let this be C}} + 2\int_1^{\frac{1-\nu}{1+\nu}} \frac{\chi_3(\nu)}{(1+\nu)^2}\mathrm{d}\nu = C \\
&- \frac{2\chi_3(\nu)}{1+\nu}\Big|_1^{\frac{1-\nu}{1+\nu}} + 2\int_1^{\frac{1-\nu}{1+\nu}} \frac{\chi_2(\nu)}{\nu(1+\nu)}\mathrm{d}\nu = C + (1-\nu)\chi_3\left(\frac{1-\nu}{1+\nu}\right) - \frac{7\zeta(3)}{8} \\
&- \quad 2\chi_2(\nu)\ln(1+\nu)\Big|_1^{\frac{1-\nu}{1+\nu}} + \int_1^{\frac{1-\nu}{1+\nu}} \frac{\ln(1+\nu)\ln\left(\frac{1+\nu}{1-\nu}\right)}{\nu}\mathrm{d}\nu = C + (1-\nu)\chi_3\left(\frac{1-\nu}{1+\nu}\right) \\
&- \frac{7\zeta(3)}{8} + 2\chi_2\left(\frac{1-\nu}{1+\nu}\right)\ln\left(\frac{1+\nu}{2}\right) + \frac{\pi^2}{4}\ln 2 + \frac{1}{2} \\
&\int_1^{\frac{1-\nu}{1+\nu}} \frac{\ln^2(1+\nu) - \ln^2(1-\nu) + \ln^2\left(\frac{1-\nu}{1+\nu}\right)}{\nu}\mathrm{d}\nu
\end{aligned}
$$

Fig. 5. MathJax rendering without enrichment.

$$
\begin{aligned}
I_\nu(\nu^{-1},1) &= \underbrace{\frac{\pi^2}{4}\ln\left(\frac{(1+\nu)^{1+\nu}}{\nu^\nu}\right) - \frac{7\,\zeta(3)}{8}\nu}_{\text{Let this be C}} + 2\int_1^{\frac{1-\nu}{1+\nu}} \frac{\chi_3(\nu)}{(1+\nu)^2}\mathrm{d}\nu \\
&= C - \frac{2\chi_3(\nu)}{1+\nu}\Big|_1^{\frac{1-\nu}{1+\nu}} + 2\int_1^{\frac{1-\nu}{1+\nu}} \frac{\chi_2(\nu)}{\nu\,(1+\nu)}\mathrm{d}\nu \\
&= C + (1-\nu)\,\chi_3\left(\frac{1-\nu}{1+\nu}\right) - \frac{7\,\zeta(3)}{8} - \quad 2\chi_2(\nu)\ln(1+\nu)\Big|_1^{\frac{1-\nu}{1+\nu}} + \int_1^{\frac{1-\nu}{1+\nu}} \frac{\ln(1+\nu)\ln\left(\frac{1+\nu}{1-\nu}\right)}{\nu}\mathrm{d}\nu \\
&= C + (1-\nu)\,\chi_3\left(\frac{1-\nu}{1+\nu}\right) - \frac{7\,\zeta(3)}{8} + 2\chi_2\left(\frac{1-\nu}{1+\nu}\right)\ln\left(\frac{1+\nu}{2}\right) + \frac{\pi^2}{4}\ln 2 \\
&\quad + \frac{1}{2}\int_1^{\frac{1-\nu}{1+\nu}} \frac{\ln^2(1+\nu) - \ln^2(1-\nu) + \ln^2\left(\frac{1-\nu}{1+\nu}\right)}{\nu}\mathrm{d}\nu
\end{aligned}
$$

Fig. 6. Rendering of semantically enriched content.

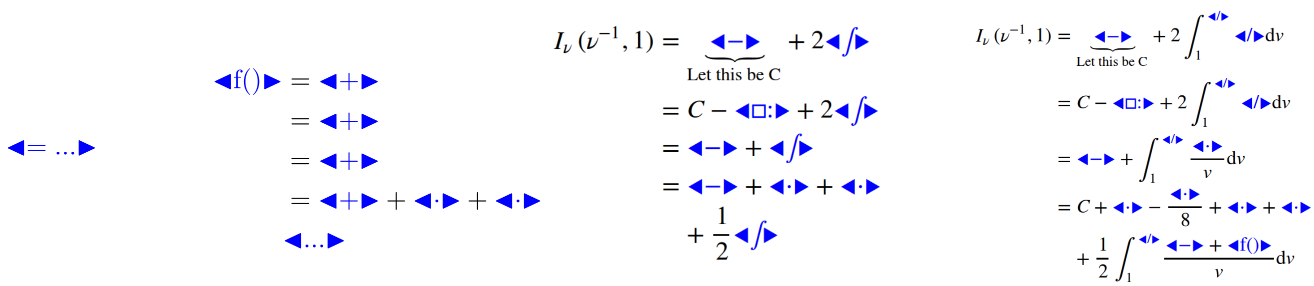

Fig. 7. Four different stats of Collapse of example equation.

Instead, it estimates the complexity of an expression by recursively evaluating the enriched MathML tree. For example, token elements are assigned a complexity value according to their string content while more complex elements such as roots or fractions are assigned the sum of their children's complexity measures modified by a value corresponding to their own visual complexity. Cut-off values for each semantic type are then used to decide which expressions are collapsed. The parameters (character weight, operator modifier, cut-off) are configurable by the content author.

VII. Aural Rendering

In a final step, we now exploit our semantic enrichment, and in particular the structural abstraction, as an accessibility service for screen readers. Currently, MathJax already provides support for third-party screen readers by embedding the MathML representation for formulas as hidden elements into the DOM. This can be picked up by a screen reader to either voice directly (e.g., in the case of ChromeVox or VoiceOver), or by using a third-party library like MathPlayer [2] for speech translation. Thus users still have to depend on their screen reader's ability to understand MathML to provide either of these two capabilities. Despite being a pragmatic solution today, this severely limits accessibility as it disconnects visual and AT rendering completely.

In order to make users independent of a screen reader's Math capabilities, MathJax now exploits direct access to the speech-rule engine [6] that generates the semantic-tree transformation in the first place and provides aural renderings of mathematical expressions that can be directly exposed to a screen reader. This feature is offered together with an interface for interactive exploration of the mathematical expression either along the syntactic structure or the semantic tree.

The real novelty in exploiting the semantic enrichment, however, is that it enables us to not only display but also voice semantically informed summaries of mathematical expressions using appropriate speech rules. This way, even complex formulas can be described in a concise manner, aiding a casual reading experience. The idea is that the reader gets an impression of what type of formula is there while reading, and can then choose to get more information or dive deeper into parts of the formula should they so desire.

The basic principle is to exploit the structure provided by the collapse/expansion mechanism implemented via the `maction` elements. Speech rules then can directly interpret the top level of the collapsed expression, effectively voicing only the summary symbol together with the rest of the equation. Alternatively, slightly more expressive speech rules can voice a summary of the underlying collapsed expression.

For example, in the case of the quadratic formula, the only collapse possible is the sum on the left-hand side of the equation. Two alternative ways of speaking it are "Sum equals 0" or "Sum with three summands equals 0."

This feature can be integrated into the expansion interface by providing standard cursor key bindings to expand step-by-step the collapsed parts of an expression while speaking the newly revealed sub-formulas. Technically, we achieve this by introducing a dedicated assertive ARIA live region into the DOM and updating it with the desired speech output. Speech strings for a formula and all its sub-expressions can be

```
<math speech="sum with three summands equals 0" complexity="7">
  <maction complexity="2" speech="sum with three summands">
    <mtext mathcolor="blue">&#x25C2;+&#x25B8;</mtext>
    <mrow speech="a x squared plus b x plus c" complexity="24.8">
      <mrow speech="a x squared" complexity="10.8">
        <mi speech="a" complexity="1">a</mi>
        <mo speech="times" complexity="1">&#x2062;</mo>
        <msup speech="x squared" complexity="5.8">
          <mi speech="x" complexity="1">x</mi>
          <mn speech="2" complexity="1">2</mn>
        </msup>
      </mrow>
      <mo speech="plus" complexity="1">+</mo>
      <mrow speech="b x" complexity="6">
        <mi speech="b" complexity="1">b</mi>
        <mo speech="times" complexity="1">&#x2062;</mo>
        <mi speech="x" complexity="1">x</mi>
      </mrow>
      <mo speech="plus" complexity="1">+</mo>
      <mi speech="c" complexity="1">c</mi>
    </mrow>
  </maction>
  <mo speech="equals" complexity="1">=</mo>
  <mn speech="0" complexity="1">0</mn>
</math>
```

Fig. 8. Embedded speech strings and complexity for the quadratic formula. Observe that other semantic attributes have been omitted.

pre-computed and embedded into the MathML along-side the semantic information as an additional data attribute, thereby exposing it also in the rendered DOM elements. This allows us to simply retrieve the string and update the ARIA live region, allowing its content to be voiced by most screen readers.

Fig. 8 depicts the enriched MathML expression with embedded speech strings for the quadratic formula. Note the summary message in the `maction` element. By default we currently employ the MathSpeak rule set [8] for embedded speech generation, with the exception of the summarising, where we have a small, newly developed set. This can be changed to other rule sets, e.g., the original ChromeVox rules.

The advantage of directly embedding speech is that it is available without lag when voicing the expression. However, speech strings are always generated, even in the case when no screen reader is present, which can become computationally expensive on pages with many mathematical expressions. In addition, it is more difficult to customise speech rules programmatically on pages for specialised mathematical content.

To avoid computational overhead and support easier customisation, an alternative is to generate speech strings on the fly, if and when necessary, and expose them via the ARIA live region. While this has the advantage that it is easy to customise rules or use different rule sets altogether — and we indeed provide a simple API that allows for adding or modifying speech rules — the recursive nature of generating speech rules can lead to computational overhead, in particular when exploring sub-expressions step-wise. As a consequence, we have implemented a caching mechanism that avoids multiple speech generations for the same sub-expressions.

VIII. Conclusions

We have presented the advanced accessibility features of MathJax that are based on a new semantic enrichment procedure for Presentation MathML. In particular, the novel approach to responsive rendering of equations shows promise for more advanced AT techniques. Although we have not yet had the opportunity to do extensive user testing, since our code is developed publicly on our GitHub repository, we have already had some user feedback. Initial reactions have been extremely positive, but we hope to pursue collaborations for wider, real-world testing in 2016.

While the presented AT extensions are not yet available in core MathJax, we plan to release a MathJax extension usable with current MathJax versions and to move the advances closer to the core during our upcoming work on MathJax v3.0. As our tools can run both client- and server-side (via NodeJS), they can be leveraged both by page authors and by end users (via bookmarklets or browser extensions), as well as directly in assistive technology.

We believe our work can be greatly expanded. First, the heuristics can be augmented to enhance precision of the analysis. Second, subject-specific rules can be implemented to improve precision; in the long term, this could enable the combination of natural-language processing tools to augment our heuristics by analysing the surrounding context. Third, we believe the semantic enrichment of MathML can inform web standards development regarding improvements to MathML and ARIA to enable other AT solutions to build on our results.

Acknowledgement

The work was supported by the Alfred P. Sloan Foundation.

References

[1] A. Karshmer, G. Gupta, and E. Pontelli, "Mathematics and accessibility: a survey," in *Proc. 9th ICCHP*, vol. 3118, 2007, pp. 664–669.

[2] N. Soiffer, "Mathplayer: web-based math accessibility," in *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 2005, pp. 204–205.

[3] D. Carlisle, P. Ion, and R. Miner, "MathML version 3.0," W3C Recommendation, 2010, www.w3.org/TR/MathML3.

[4] Wide Web Consortium, "HTML version 5.0," World W3C Candidate Recommendation, 2013, available at www.w3.org/TR/html5.

[5] M. Consortium, "MathJax version 2.5," 2014, www.mathjax.org.

[6] V. Sorge, C. Chen, T. Raman, and D. Tseng, "Towards making mathematics a first class citizen in general screen readers," *W4A 2014*. ACM.

[7] "Apple voiceover," https://www.apple.com/accessibility/osx/voiceover/.

[8] A. Nemeth, "Mathspeak," www.gh-mathspeak.com/, 2005.

[9] Maplesoft, "Documentation Center: MathML package," 2012. http://www.maplesoft.com/support/help/Maple/view.aspx?path=MathML.

[10] Wolfram Language, "Working with MathML," 2012, http://reference.wolfram.com/mathematica/XML/tutorial/MathML.html.

[11] B. Miller, "LateXML: A LaTeX to XML converter," 2010. http://dlmf.nist.gov/LaTeXML

[12] R. Moore, "Semantic enrichment of mathematics via 'tooltips'," in *Proc. of CICM 2015*, LNCS 9150, Springer, 2015.

[13] D. McKain, "Snuggletex v1.2.2," University of Edinburgh, 2010, http://www2.ph.ed.ac.uk/snuggletex/

[14] www.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/.

[15] www.smashingmagazine.com/2014/03/05/rethinking-responsive-svg/.

[16] zurb.com/playground/projects/responsive-tables/index.html.

[17] L. Rello and R. Baeza-Yates, "Optimal colors to improve readability for people with dyslexia," in *Text Customization for Readability*, 2012.

[18] R. Jones, "Strategies for reading comprehension: Selective underlining," 2008.

[19] "Usage share of web browsers," https://en.wikipedia.org/wiki/Usage_share_of_web_browsers#Summary_tables.