Tagged Mathematics in PDFs for Accessibility and other purposes

Ross Moore

Mathematics Department, Macquarie University, Sydney, Australia
ross.moore@mq.edu.au

Abstract. PDF has been the preferred format for publishing mathematics for many years now. With changes to methods of delivery (i.e., electronic rather than predominantly paper) there need to be corresponding enhancements in the document format. Not least among these can be implicit legal obligations to satisfy Accessibility criteria.

The answer developed for PDF is tagging of document structure and content types, as described in the PDF/UA Implementation Guide [4]. Wikipedia describes this as "not a separate file-format but simply a way to use PDF" [12], which when supported "reader software will be able to reliably reflow text onto small screens, provide powerful navigation options, transform text appearance, improve search engine functionality, aid in the selection and copying of text, and more" [12]. Academic publishers are starting to see these benefits and will doubtless soon require at least minimal tagging of online PDF documents for Accessibility purposes, in a similar way to how Accessibility tags have been incorporated into HTML.

Here is a brief overview of work done by the author to incorporate full MathML tagging of mathematical content in documents produced primarily using the LaTeX typesetting system. Since the publicly available TeX software was not written to support such tagging of document content, further software tools are also required. This includes using a modified version of pdfTeX, a self-developed Perl program, TeX to MathML conversion software, some standard Unix command-line utilities, and extensive use of self-written TeX and LaTeX macros.

As this work is a continuation of work presented at the CICM meetings in 2009 [5], we concentrate here mostly on the advancements made since then. This includes the ability to capture complete math-environments from a running LaTeX job, to automatically invoke a conversion of the LaTeX source of the particular piece of mathematics into Presentation MathML using whatever appropriate conversion software is available. Previously the MathML version needed to have been available independent from the LaTeX source. Now this conversion can be done 'on-the-fly', using TeX4HT for example, before merging the MathML and LaTeX descriptions of the same piece of mathematics into a new extended LaTeX description incorporating macros to cause the generation of appropriate tagging and enrichment to satisfy Accessibility requirements. Such automatic conversion and merging can add significantly to the total running time for the whole job, so an indexing system has been developed which

allows the resulting enriched LaTeX description to be reused with multiple occurrences of the same source coding within the same job, and to be available for reuse in subsequent LaTeX runs.
Another development is better control over the words produced for alternative text, to be read by screen-readers. Where previously this was largely hard-coded in the enriched LaTeX description, this is now replaced by macros whose expansion text can be customised. This allows for the possibility of generating speech text in different languages, or customising what is to be spoken according to the field of mathematics being described within the document. Such customisations can be done at the LaTeX level, so that a document author need not be involved with the highly intricate details of conversion to MathML and the enhancements required for tagging.

1 Background, Overview of "Tagged PDF"

The Web Content Accessibility Guidelines (WCAG 2.0) [9], have been developed primarily for the construction of websites to allow easier access to, and navigation of, online content by persons having a disability, in particular by people with visual impairment. The governments of some countries [7], including Canada [11] and Australia [10], have established that adherence to these guidelines be a requirement, at least for their own governmental web presence.

While PDF files are not websites, the same principles of Accessibility should still apply [4], which make tagging of both structure and content within a PDF document an issue of some importance. Indeed the latest version of Adobe's 'Acrobat Pro' application has a suite of 32 checks which test whether a document meets various aspects of the WCAG recommendations [8]. Typically a PDF produced using LaTeX software is lucky to satisfy 11 of these checks, with two others requiring manual verification anyway. Lucky, because some of these checks refer to content/structure types not even present, hence not deemed to be failing.

With the help of a modified version of pdfTeX, having extra primitives specifically to enable production of proper "Tagged PDF" documents, the author has been able to produce PDFs containing extensive technical and mathematical content, that fail none of those 32 tests. Such documents are deemed properly Accessible by Adobe's checks, while at the same time being capable of validation against both the PDF/A-2b and PDF/A-2u [4] specifications. Just using the modified version of pdfTeX is not sufficient. While this enables tagging to be implemented, much macro programming is required to (i) load extra Metadata and font-encoding mappings; (ii) provide alternative-text to be read by screen-reader software; (iii) organise how the new tagging primitives are used and are correctly nested; (iv) identify structural "artifacts" (such as page-numbers, footnote rules, background images, etc.); (v) keep track of aspects of the document structure; and (vi) many other details requiring special attention. Some of these tasks enhance the Accessibility without the need for tagging; but the tagging is

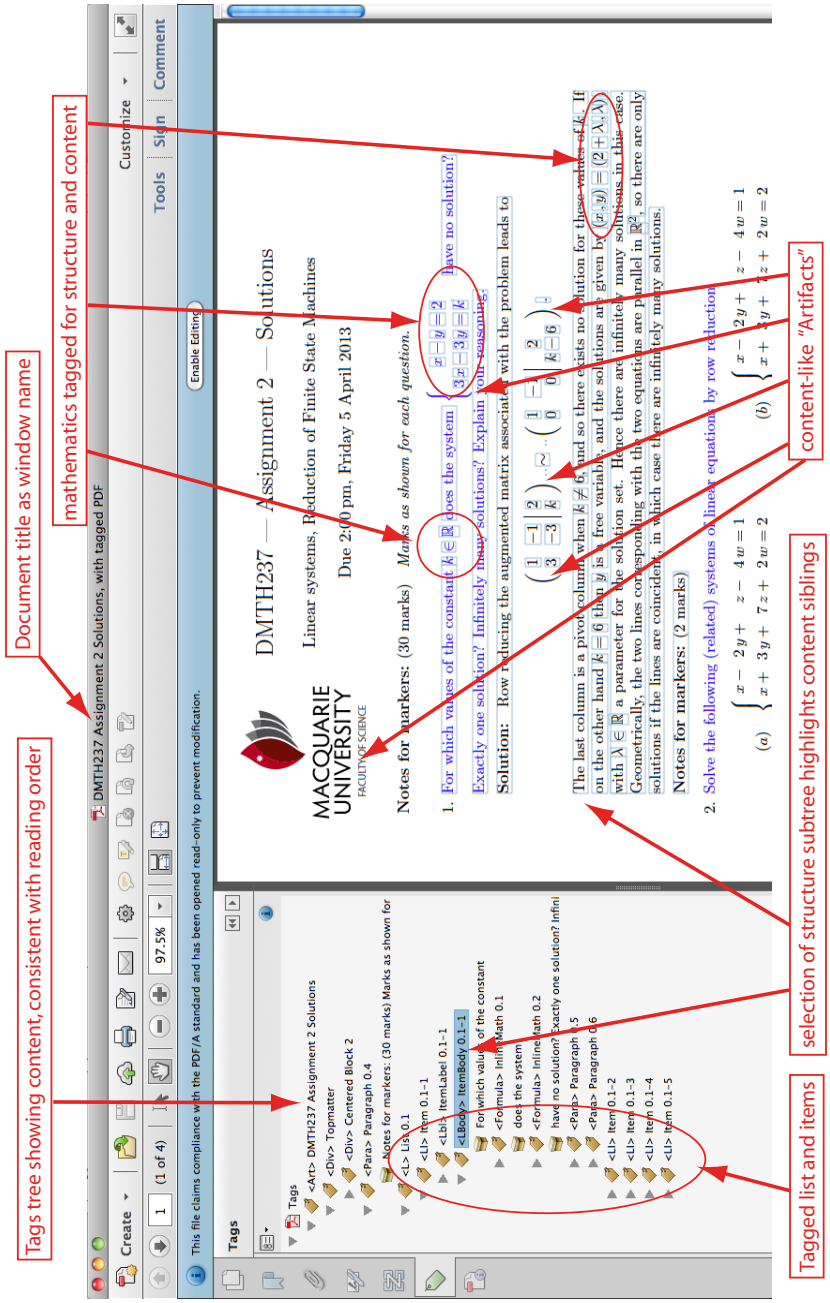

Fig. 1. View of the 'Tags' tree as seen in Acrobat Pro XI, for a typical document containing mathematical content, indicating some aspects particularly relevant to Accessibility considerations. Light blue boxes in the main viewing panel indicate separately tagged portions of content.

certainly the most important requirement. Figure 1 shows part of a document, indicating some of the above-listed features.

Now the WCAG recommendations do not specify the detail to which mathematical formulas should be tagged, but the natural format would be Presentation MathML. Furthermore, since the forthcoming ISO 32000-2 specifications are to include MathML tagging, this is the natural choice for handling mathematical content. In subsequent sections, we first give a brief overview of how tagging is implemented in PDF, then show what this means for a simple piece of mathematical content, in which each single character has a special meaning and fits within a very tight structural description. Also shown, in Figure 5 is the enriched LaTeX source to achieve this, requiring many lines of input to capture the meaning and structure of just a few characters in the original LaTeX description.

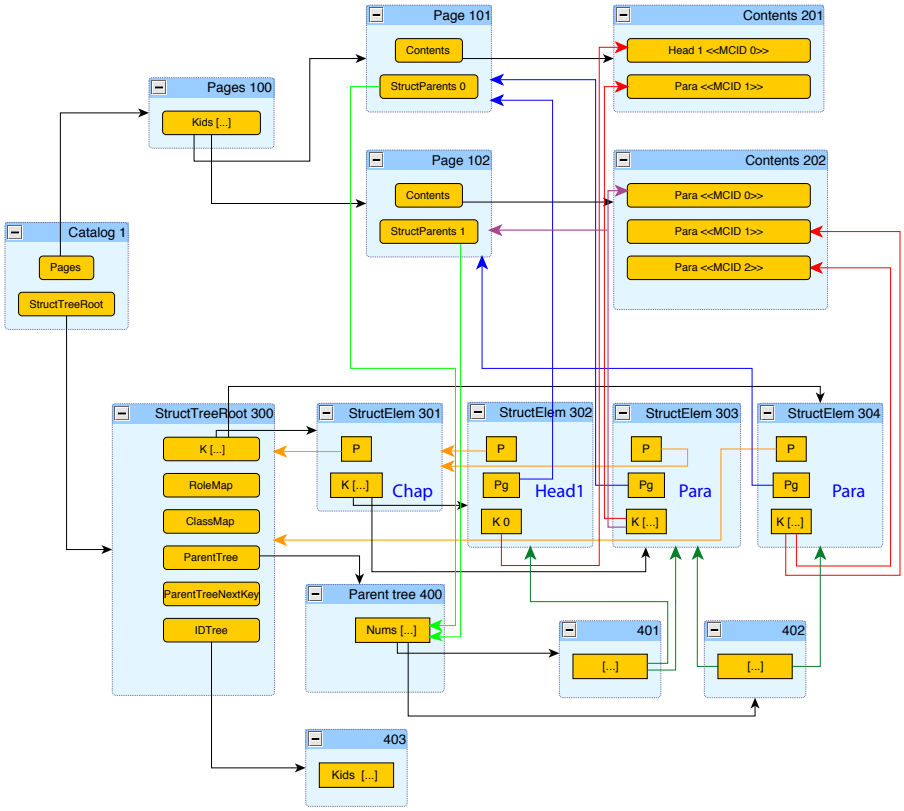

Fig. 2. Interleaving of structure and content tagging within a 2-page PDF document, structured as a heading and two paragraphs. (based on an example in [1]). Picture reproduced with permission from [5].

2 Overview of "Tagged PDF"

A brief history of Adobe's PDF format and specification was given by the author at the CICM 2009 meeting, and published as [5]. Rather than repeat this here, the basic structure of "Tagged PDF" can be understood using the diagram, reproduced from [5] and shown as Figure 2. This has been described as a "double-tree" structure. The 'Pages' tree is indicated by the top rows of blue boxes, headed as 'Page ...', each having another box headed as 'Contents ...' which denotes the 'page content' streams; that is, the low-level commands to select fonts and place text on the page. Relevant documentation and format specifications are listed in the bibliography, as [1,2,3,4,8].

The 2nd tree is the 'Structure Tree' which is much richer, represented by the lower rows of blue boxes. Leaf-nodes for this tree consist of the tags show as yellow rectangles within the blue 'Contents ...' boxes, each with labels including 'MCID ...'. All the other yellow rectangles represent data structures required to do the 'book-keeping' of how these two tree structures intersect. In an actual Tagged PDF document, these allow for navigation and selection of the content using the 'structure elements', as shown in Figure 1, and for extraction of individual pages so as to preserve that part of the structure that is present on the specific page.

A useful way to see these two tree structures together is provided by the 'Tags' view, in Adobe's Acrobat Pro software. See Figure 1 for an example of how this can appear.

3 Example of Tagged content within a 'page stream'

As with XML and HTML, tags can have attributes to help reader software tools construct a more appropriate representation of the content. Two such attributes are the '/Alt' text and '/ActualText' character strings. The former is used to provide a short textual description of the content of an image, just as with HTML for web-pages. But this attribute is available for any content, so is particularly useful with mathematics to provide an audible description of the names of symbols or their uses. On the other hand, the '/ActualText' is used with text-extraction, to provide a mapping from characters shown onscreen to a friendlier representation; e.g., the old German ß character may be mapped to 'ss', or to the Unicode point U+00DF when the font used in the document is coded in a non-standard way. For mathematical symbols, this can be used to map variable names x, y, a, b, etc. into mathematical alpha-numerics, such as U+1D465 etc., when the document font uses ASCII positions, as in the following example.

A portion of the page stream corresponding to part of the content which renders as $\left(\frac{4}{3}\pi R^3\right)$ is shown in Figure 3. The lines which contain '/MCID' and those following up to 'EMC' are what determine the tagging of content. All other lines are as would appear without tagging, apart from line-ends being removed to allow this code sample to fit on the printed page. Notice the use of hexadecimal strings for '/ActualText' and plain English words for '/Alt' text, which are the 'Accessible Text' words that will be vocalised by screen-reader software. The '/mi', '/mn' clearly corresponds to MathML tagging of the content. For this example the MathML description used is shown in Figure 4.

```
1 0 0 1 301.237 0 cm
/Artifact << /Type/Layout/ActualText<FEFF0028> >>BDC
1 0 0 1 1.992 11.557 cm
BT /F27 9.9626 Tf [(\020)]TJ ET EMC
1 0 0 1 7.146 -6.849 cm
/mn <</MCID 6 /ActualText<FEFF0034>/Alt(  : open bracket: four  )>>BDC
BT /F19 7.9701 Tf [(4)]TJ ET EMC
q 1 0 0 1 0 -1.719 cm
[]0 d 0 J 0.398 w 0 0 m 4.234 0 l S
Q 1 0 0 1 0 -8.83 cm
/mn <</MCID 7 /ActualText<FEFF0033>/Alt(  thirds )>>BDC
BT /F19 7.9701 Tf [(3)]TJ ET EMC
1 0 0 1 5.43 4.122 cm
/mo <</MCID 8 /ActualText<FEFF2062>/Alt(  times  )>>BDC
BT /F76 1 Tf [( )]TJ ET EMC
/mi <</MCID 9 /ActualText<FEFF03C0>/Alt(  pi  )>>BDC
BT /F21 11.9552 Tf  [(\031)]TJ ET EMC
1 0 0 1 7.069 0 cm
/mi <</MCID 10 /ActualText<FEFFD835DC45>/Alt(  capital R  )>>BDC
BT /F21 11.9552 Tf [(R)]TJ ET EMC
1 0 0 1 9.008 4.339 cm
/mn <</MCID 11 /ActualText<FEFF0033>/Alt(  cubed,  close bracket: )>>BDC
BT /F19 7.9701 Tf [(3)]TJ ET EMC
```

Fig. 3. Portion of the PDF page-stream contents for a piece of inline mathematics: $\left(\frac{4}{3}\pi R^3\right)$. White-space, as line-ends, has been massaged for convenience of display. The 'Accessible Text' as read by a screen reader is visible in the /Alt attributes. This combines to produce ' : open bracket: four thirds times pi capital R cubed, close bracket: ', with each ':' character inducing a slight pause. Font characters are accompanied by /ActualText replacements which allow copy/paste to provide correct Unicode points. Brackets must be tagged as /Artifact since they do not correspond to tagged content within the MathML description (see Figure 4); words to be read are shifted to the nearest enclosed content.

```
<math xmlns="http://www.w3.org/1998/Math/MathML"
   style="font-size: xx-small">
  <mfenced separators="">
    <mfrac> <mn>4</mn> <mn>3</mn> </mfrac>
    <mo>&#x2062;<!-- &InvisibleTimes; --></mo>
    <mi>&#x03C0;<!-- &pi; --></mi>
    <msup> <mi>R</mi> <mn>3</mn> </msup>
  </mfenced>
</math>
```

Fig. 4. MathML representation of the same piece of mathematics: $\left(\frac{4}{3}\pi R^3\right)$, used to create the PDF portion shown in Figure 3. White-space has been massaged for display purposes.

```
\SMC{attr{/Type/Layout/ActualText<FEFF0028>} }{-1}{Artifact}%
 \left(%  start of fence
   \EMA
   \TPDFfbrack
 \tfrac{
\pdfinterwordspaceoff
   \SMC{attr{/ActualText<FEFF0034>\TPDFaloud{0034}} }{5}{mn}%
 4%
   \EMC
 }{%
   \SMC{attr{/ActualText<FEFF0033>\TPDFspeak{\TPDFthirds}} }{6}{mn}%
 3%
   \EMC
 }%
   \SMC{attr{/ActualText<FEFF2062>\TPDFaloud{2062}} }{7}{mo}%
 \pdffakespace
   \EMC
   \SMC{attr{/ActualText<FEFF03C0>\TPDFaloud{03C0}} }{8}{mi}%
 \pi%
   \EMC
   \SMC{attr{/ActualText<FEFFD835DC45>\TPDFaloud{1D445}} }{10}{mi}%
 R%
   ^{%
   \EMC
   \TPDFcubed
   \TPDFpopfence
   \SMC{attr{/ActualText<FEFF0033>\TPDFaloud{0033}} }{11}{mn}%
 3%
   \EMC
 }%
   \SMC{attr{/Type/Layout/ActualText<FEFF0029>} }{-1}{Artifact}%
 \right)%  end of fence
   \EMA
\SSE{attr{/Type/StructElem/S/math
/A<</O/XML-1.00/xmlns(http://www.w3.org/1998/Math/MathML)
  /style(font-size: xx-small)>>}\TPDFaloudtag{math}{2}}{1}{0}{2}%
\SSE{attr{/Type/StructElem/S/mfenced
/A<</O/XML-1.00/separators()>>}\TPDFaloudtag{mfenced}{3}}{2}{2}{3}%
\SSE{attr{/Type/StructElem/S/mfrac}\TPDFaloudtag{mfrac}{4}}{1}{3}{4}%
\SSE{attr{/Type/StructElem/S/mn}\TPDFaloudtag{mn}{5}}{2}{4}{5}%
\SSE{attr{/Type/StructElem/S/mn}\TPDFaloudtag{mn}{6}}{3}{4}{6}%
\SSE{attr{/Type/StructElem/S/mo}\TPDFaloudtag{mo}{7}}{2}{3}{7}%
\SSE{attr{/Type/StructElem/S/mi}\TPDFaloudtag{mi}{8}}{3}{3}{8}%
\SSE{attr{/Type/StructElem/S/msup}\TPDFaloudtag{msup}{9}}{4}{3}{9}%
\SSE{attr{/Type/StructElem/S/mi}\TPDFaloudtag{mi}{10}}{2}{9}{10}%
\SSE{attr{/Type/StructElem/S/mn}\TPDFaloudtag{mn}{11}}{3}{9}{11}%
\TPDFcleanread {11}%
```

Fig. 5. Output from texmmljoin merging TEX source with the MathML content from Figure 4. The original LATEX source can be read down the left-hand edge, as `\left (\frac 43 \pi R^3 \right )`. The lower portion builds the structure tree, using tag indices which are interpreted relative to an offset.

```
39 0 obj << /K [ 34 0 R 35 0 R ] /P 37 0 R
  /Type/StructElem/S/msup >> endobj
38 0 obj << /K [ 29 0 R 30 0 R ] /P 37 0 R
  /Type/StructElem/S/mfrac >> endobj
37 0 obj << /K [ 38 0 R 31 0 R 32 0 R 39 0 R ] /P 36 0 R
  /Type/StructElem/S/mfenced /A<</O/XML-1.00/separators()>> >> endobj
36 0 obj << /K [ 37 0 R ] /P 27 0 R
  /Type/StructElem/S/math /A<</O/XML-1.00
   /xmlns(http://www.w3.org/1998/Math/MathML)
   /style(font-size: xx-small)>> >> endobj
35 0 obj << /K [ 11 ] /Pg 5 0 R /P 39 0 R
  /Type/StructElem/S/mn >> endobj
34 0 obj << /K [ 10 ] /Pg 5 0 R /P 39 0 R
  /Type/StructElem/S/mi >> endobj
32 0 obj << /K [ 9 ] /Pg 5 0 R /P 37 0 R
  /Type/StructElem/S/mi >> endobj
31 0 obj << /K [ 8 ] /Pg 5 0 R /P 37 0 R
  /Type/StructElem/S/mo >> endobj
30 0 obj << /K [ 7 ] /Pg 5 0 R /P 38 0 R
  /Type/StructElem/S/mn >> endobj
29 0 obj << /K [ 6 ] /Pg 5 0 R /P 38 0 R
  /Type/StructElem/S/mn >> endobj
27 0 obj << /K [ 36 0 R ] /P 26 0 R
  /Type/StructElem/S/Formula /ID(Math0.1)/T(InlineMath 0.1)
 /A<</O/XML-1.01 /TeX(sphere_volume-1.tex) /MathML(sphere_volume-1.txt)>>
>> endobj
```

Fig. 6. Portion of the PDF structure tree for a piece of inline mathematics with MathML tagging as in Figure 4, corresponding to the bottom output lines from texmmljoin as shown in Figure 5. White-space has been massaged for convenience of display. Each structure node is represented as a PDF dictionary object with keys for the '/Type', structure-type '/S', any attributes '/A', parent '/P', page reference '/Pg' and '/Kids' array. The latter contains either the '/MCID' numbers for the leaf-node 'marked content', as seen in Figure 3, and/or object-references for any sibling structure-nodes, listed in order within the structure tree. The root-node of this portion of the structure tree is a '/Formula', whose parent is the surrounding paragraph. This has an index '/ID' and title '/T' which can be used by PDF browser clients for lists of the structure, and extra non-standard tags '/TeX' and '/MathML' giving the local names of the files which were used to produce the merged content of Figure 5.

4 Encoding the tagging with LaTeX

Starting from the LaTeX source `$\left (\frac 43 \pi R^3 \right )$`, and a MathML description obtained using TeX-to-MathML conversion software, as shown in Figure 4, these two descriptions which capture the same mathematical content need to be merged. This is done using a Perl program, written by the

author, called texmmljoin. For the specific code sample its output includes the TEX-like coding shown in Figure 5.

The original LATEX source for the mathematics can be seen down the left-hand edge. Also, '/ActualText' attributes are clearly seen, but what about the '/Alt' attribute? This is built from the \TPDFaloud, \TPDFaloud and other special command sequences such as \TPDFfbrack, \TPDFthirds, \TPDFcubed and \TPDFpopfence. Using TEX macro coding, \TPDFaloud{1D445} expands into something specific to the unicode point U+1D445 which is an uppercase (or 'capital') letter 'R'. This is what is spoken in English, but that could be replaced with whatever is appropriate for another spoken language, simply using macro definitions — the output from texmmljoin does not need to be changed. Macros \TPDFcubed and \TPDFpopfence add words to become part of the '/Alt' attribute for the subsequent piece of content, or the previous when there is no more. These also may be customised for languages other than English, or to change what is to be spoken according to the mathematical context of their use. For example, {...} might be notation for a 'set' in one piece of mathematics, but in another it could be denoting a 'Lie bracket'. Through macro definitions, the spoken version can be adapted to provide the appropriate words.

The lower part of the output from texmmljoin, as shown in Figure 5, contains information on how to build the structure tree. These have the form of a TEX macro (\SSE) taking four arguments, the first indicating the MathML tag type along with any attributes, with final three being bracketed numbers. The last of these is a unique index for the structure tag, with the preceding one being the index of its parent structure tag. Before these, the first bracketed number is an index which affects the ordering of those structure tags having the same parent; lower numbers occur earlier within the structure tree. The PDF objects that record this information within the PDF file are shown in Figure 6.

For a specific math-environment the parent and structure tag indices as shown here are not used directly, but are first incremented by an *offset* determined as the environment is encountered. This offset is essentially the highest tag index encountered so far, ensuring that the index used is unique for all structure tags within the same document, allowing the structure tree to be faithfully constructed internally by pdfTEX. The numerical argument to \TPDFcleanread conveys the total number of new structure tags defined within the math-environment, so that the offset can be incremented correctly for subsequent tagged content.

5 Automatic generation of MathML

Previous work done by the author [6] detected all the math-environments in a LATEX document, writing the contents into an external file before creating button annotations to show/hide textual fields displaying these original source code snippets. This work has now been adapted to prepare source for TEX to MathML translation software; e.g., based upon TEX4HT. Thus now an existing LATEX document can be processed, detecting the math-environments and initiating a

conversion to MathML. Then texmmljoin is run to merge the resulting pair of files having LaTeX and MathML descriptions of the same piece of mathematics, creating new LaTeX source enriched with full structure and content tagging.

Since TeX4HT needs a complete run of LaTeX on the source snippet, followed by further post-processing to generate an XML file containing the MathML description of the mathematics, this process can typically take many seconds. To avoid repeating this work on each run of LaTeX over the document source, an indexing mechanism has been developed, which associates the specific source coding with the name prefix of the files created in the translation and merging processes. Having created enriched source for a piece of mathematics, there is no need to redo it, unless the content is changed by edited. Due to the use of tag index offsets, as described in Section 4, editing which simply reorganises the order of appearance of pieces of mathematics does not need to initiate re-tagging. Using a different offset, the internal index remains unique for all structure tags.

```
<file>2013-Assign2-soln-inline-1</file><code>\( k \in \RR \) </code>
<file>2013-Assign2-soln-inline-2</file><code>\( \begin {cases}\begin
 {aligned} x - y & = 2 \\ 3\,x - 3\,y & = k \end {aligned}\end {cases}
  \) </code>
<file>2013-Assign2-soln-display-1</file><code>\[ \left ( \begin {array}
 {rr | c} 1 & -1 & 2 \\ 3 & -3 & k \end {array} \right ) \;\; \sim \;\;
 \left ( \begin {array}{rr | c} 1 & -1 & 2 \\ 0 & 0 & k-6 \end {array}
  \right )\,. \] </code>
<file>2013-Assign2-soln-inline-3</file><code>\( k \not = 6 \) </code>
<file>2013-Assign2-soln-inline-4</file><code>\( k \) </code>
<file>2013-Assign2-soln-inline-5</file><code>\( k=6 \) </code>
<file>2013-Assign2-soln-inline-6</file><code>\( y \) </code>
<file>2013-Assign2-soln-inline-7</file><code>\( (x\,,y)= (2+\lambda
 \,, \lambda ) \) </code>
<file>2013-Assign2-soln-inline-8</file><code>\( \lambda \in \RR \)
 </code>
<file>2013-Assign2-soln-inline-9</file><code>\( \RR ^2 \) </code>
```

Fig. 7. Portion of the index of math content seen in Figure 1, with line-breaks massaged for readability.

One further advantage of this is that if the same piece of mathematics is used repeatedly within a document, the generation and merging with MathML need only be performed once. The previous output from texmmljoin may simply be reused with different offsets ensuring that the resulting PDF document is correctly formed. Thus much time-consuming work needs to be done only once. Furthermore the way that the indexing is constructed, making use of the `\detokenize` primitive of e-TeX and scanning to reduce runs of multiple spaces to a single token, means that nearly identical math environments can be treated as being actually identical for tagging purposes. The index of the mathematical

content is written to an external file, which is then read at the beginning of the next LaTeX run. This allows the association to be made early between a piece of mathematics and the files required for its enrichment. Figure 7 shows a portion of the index file, used with the document seen in Figure 1.

Of course the tagged math-environments have to be fitted into the structure tagging of the document as a whole, generally as siblings of the surrounding paragraph, which may or may not finish with the mathematics. Details of the TeX coding to achieve this is beyond the scope of this report. This should be discussed in a later report, delivered perhaps at a different forum.

References

1. Adobe Systems Inc.; PDF Reference 1.7, November 2006. http://www.adobe.com/devnet/pdf/pdf_reference.html. Also available as [3].
2. ISO 19005-1:2005; Document Management — Electronic document file format for long term preservation — Part 1: Use of PDF 1.4 (PDF/A-1); http://www.iso.org/iso/catalogue_detail?csnumber=38920.
3. ISO/DIS 32000; Document management — Portable document format (PDF 1.7). July 2008. http://www.iso.org/iso/catalogue_detail?csnumber=51502.
4. Technical Implementation Guide; AIIM Global Community of Information Professionals. http://www.aiim.org/Research-and-Publications/standards/committees/PDFUA/Technical-Implementation-Guide.
   Also available as ISO 14289-1:2012; http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54564.
5. Moore, Ross R.; Ongoing efforts to generate "tagged PDF" using pdfTeX, in *DML 2009, Towards a digital Mathematics Library, Proceedings*, Petr Sojka (editor), Muni Press, Masaryk University, 2009. ISBN 978-80-20-4781-5.
   Reprinted as: TUGboat, Vol 30, No. 2 (2009), pp. 170–175.
   http://www.tug.org/TUGboat/tb30-2/tb95moore.pdf.
6. Moore, Ross R.; serendiPDF, with searchable math-fields in PDF documents; TUGboat, Vol 23, No. 1 (2002), pp. 65–69.
   http://www.tug.org/TUGboat/tb23-1/moore.pdf.
7. Policies Relating to Web Accessibility; W3C Web Accessibility Initiative. http://www.w3.org/WAI/Policy/.
8. Achieving WCAG 2.0 with PDF/UA; AIIM Global Community of Information Professionals. http://www.aiim.org/Research-and-Publications/standards/committees/PDFUA/WCAG20-Mapping.
9. Web Content Accessibility Guidelines (WCAG) 2.0. W3C Web Content Accessibility Guidelines Working Group. http://www.w3.org/TR/WCAG20/.
   Also available as ISO/IEC 40500:2012; http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=58625.
10. Statement about WCAG 2.0 compliance, in accordance with the *Disability Discrimination Act 1992*. http://australia.gov.au/about/accessibility.
11. Guidance on Implementing the Standard on Web Accessibility; Treasury Board of Canada Secretariat.
    http://www.tbs-sct.gc.ca/ws-nw/wa-aw/wa-aw-guid-eng.asp
12. Wikipedia entry for 'PDF/Universal Accessibility (PDF/UA)'. Quotations taken from: http://en.wikipedia.org/wiki/PDF/UA#Description and http://en.wikipedia.org/wiki/PDF/UA#Audience_.26_Benefits.