LaTeX conversion into normalized forms and speech

Eitan M. Gurari
Ohio State University
gurari (at) cse dot ohio-state dot edu
http://www.cse.ohio-state.edu/~gurari

Abstract

LaTeX is an authoring language designed for producing documents through native TeX compilers. Over the years many other applications have been developed to accept LaTeX inputs via alternative engines programmed from scratch. These engines are restricted in power to subsets of LaTeX features.

The first part of this report shows how TeX4ht can translate general LaTeX constructs into the restricted dialects recognizable by such engines. The jsMath dialect for rendering LaTeX through JavaScript is employed as an example.

An especially significant use of LaTeX input was T. V. Raman's 1994 pioneering AsTeR program for automatically rendering technical documents into audio. Newer audio browsers are expected to address XML documents that adhere to the SSML and ACSS specifications. The second part of this report extends Raman's work by showing how TeX4ht can translate LaTeX to XML-based representations that support speech.

1 Applications of LaTeX dialects

The LaTeX system offers a rich set of high-level features for authoring manuscripts, and a powerful engine for typesetting documents. The human friendly design of the language, in particular within its mathematical component, promoted different programs to choose variants of LaTeX as their input languages. Similarly, the superior typesetting capabilities encouraged different tools to offer LaTeX for exported document formats.

For instance, the *jsMath* utility [2] is dedicated to rendering restricted LaTeX mathematical expressions embedded within HTML files. In doing so it offers a friendly medium for on-line content management. Specifically, information is easy to enter and edit, a single document file provides for both content rendering and editing, document files can be accessible throughout the web as is the case for Wiki pages, and viewers need not install new software. The program is written in JavaScript. Figure 1(a) shows the jsMath source code for obtaining the output in Figure 1(b).

As another example, the source mathematical code of *MediaWiki* [10] is expressed in LaTeX. The code is channeled to the *texvc* program [17] for converting the expressions into images.

This material is based upon work supported by the National Science Foundation under Award No. IIS-0312487. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

On the other hand, the *Scientific Notebook* document processing system [15] is an example of a utility capable of exporting LaTeX documents. The LaTeX mathematical code emitted by this program can be imported into the *Duxbury Braille Translator* for embossing the expressions into Nemeth braille [3].

In all of the above examples, only subsets of the LaTeX features are supported. In the first two examples, minor non-LaTeX features are added.

2 A TeX4ht mode for jsMath

The jsMath system supports only a few core features of LaTeX, and its vocabulary is quite restricted due to the very limited macro capabilities of the system. TeX4ht, on the other hand, is a highly configurable converter for TeX-based sources [4]. Hence, TeX4ht can assume the bulk of the work of processing given files into forms jsMath can handle. To translate a LaTeX file named `file.tex` into HTML, with the mathematical expressions converted into jsMath, one can issue the following command:

```
htlatex file "html,jsmath" " -cmozhtf"
```

The jsMath engine recognizes a limited set of symbol names, but it fully supports Unicode representations. The flag '`-cmozhtf`' requests Unicode encodings for the majority of the symbols usually contributed from the (LA)TeX fonts, ignoring the possibility of using names for the symbols recognized by the jsMath engine.

Figure 2(a) shows the jsMath output of TeX4ht for the source of Figure 2(b), and Figure 4(a) ex-

```
<p> A quadratic equation
     <span class="math">
        ax^2 + bx + c = 0
     </span>
  with
     <span class="math">a \neq 0</span>
  has the following solution.
</p>
<div class="math">
   x = \frac {-b \pm \sqrt{b^2 - 4ac} }
             {2a}
</div>
```

(a)

A quadratic equation $ax^2 + bx + c = 0$ with $a \neq 0$ has the following solution.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

(b)

Figure 1: HTML code with embedded jsMath expressions and its rendering.

```
<p class="noindent">
  A quadratic equation
    <span class="math">
       a{x}^{2} + bx + c = 0
    </span>
  with
    <span class="math">
       a\mathrel{&#x2260;}0
    </span>
  has the following solution.
</p>
<div class="math">
    x ={ &#x2212;b &#x00B1;\sqrt{{b}^{2 }
         &#x2212; 4ac} \over 2a}
</div>
```

(a)

```
A quadratic equation $ax^2 + bx + c = 0$
with $a \ne 0$ has the following solution.

$$x={-b \pm \sqrt{b^2 - 4ac} \over 2 a}$$
```

(b)

Figure 2: TEX4ht jsMath output for a LATEX source.

```
<p class="noindent">
  A quadratic equation
     <span class="math"> a{x}^{2} + bx + c = 0 </span>
  with
     <span class="math"> a\ne 0 </span>
  has the following solution.
</p>
<div class="math">
     x =\frac{ -b\pm \sqrt{{b}^{2 } - 4ac} }
              {2a}
</div>
```

Figure 3: Reconfigured TEX4ht output.

hibits the jsMath code created for the source of Figure 4(b).

3 A taste of the TEX4ht configurations

The default jsMath configurations of TEX4ht do not take advantage of the full range of the LATEX features permitted by the jsMath utility. As a result, the jsMath code created by TEX4ht has room for improvements with respect to making the code more friendly for handling by human beings. This section demonstrates how TEX4ht can be reconfigured to produce from the input of Figure 2(b) the output of Figure 3, as an alternative to the default output shown in Figure 2(a).

3.1 Using literal characters instead of Unicode values

When LATEX encounters the minus character '-' in

```
<div class="math">
W(&#x03A6;) = \left \Vert \array{
{    &#x03C6;
\over ({&#x03C6;}_{1},{&#x025B;}_{1})}
      &amp;     0
      &amp;\mathop{\mathop{&#x2026;}}
           \kern 1.66702pt
      &amp;     0
\cr
{  &#x03C6;{k}_{n2}
\over ({&#x03C6;}_{2},{&#x025B;}_{1})}
     &amp;{     &#x03C6;
\over ({&#x03C6;}_{2},{&#x025B;}_{2})}
     &amp;\mathop{\mathop{&#x2026;}}
          \kern 1.66702pt
     &amp;     0
\cr
  .&amp;.&amp;.&amp;.&amp;.
\cr
{  &#x03C6;{k}_{n1}
\over ({&#x03C6;}_{n},{&#x025B;}_{1})}
     &amp;{   &#x03C6;{k}_{n2}
\over ({&#x03C6;}_{n},{&#x025B;}_{2})}
     &amp;\mathop{\mathop{&#x2026;}}
          \kern 1.66702pt
     &amp;{   &#x03C6;{k}_{n\kern
                 1.66702pt n&#x2212;1}
\over ({&#x03C6;}_{n},
             {&#x025B;}_{n&#x2212;1})}
     &amp;{      &#x03C6;
\over ({&#x03C6;}_{n},{&#x025B;}_{n})} }
\right \Vert
</div>
```

(a)

```
\documentclass{article}
  \usepackage{amsmath}
\begin{document}
  \[W(\Phi)= \begin{Vmatrix}
  \dfrac\varphi
        {(\varphi_1,\varepsilon_1)}&
0&\dots&0\\
  \dfrac{\varphi k_{n2}}
        {(\varphi_2,\varepsilon_1)}&
  \dfrac\varphi
        {(\varphi_2,\varepsilon_2)}
        &\dots&0\\
  \hdotsfor{5}\\
  \dfrac{\varphi k_{n1}}
        {(\varphi_n,\varepsilon_1)}&
  \dfrac{\varphi k_{n2}}
        {(\varphi_n,\varepsilon_2)}&\dots&
  \dfrac{\varphi k_{n\,n-1}}
        {(\varphi_n,\varepsilon_{n-1})}&
  \dfrac{\varphi}
        {(\varphi_n,\varepsilon_n)}
  \end{Vmatrix}\]
\end{document}
```

(b)

$$W(\Phi) = \left\Vert \begin{array}{ccccc} \frac{\varphi}{(\varphi_1,\varepsilon_1)} & 0 & \dots & 0 & \\ \frac{\varphi k_{n2}}{(\varphi_2,\varepsilon_1)} & \frac{\varphi}{(\varphi_2,\varepsilon_2)} & \dots & 0 & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\varphi k_{n1}}{(\varphi_n,\varepsilon_1)} & \frac{\varphi k_{n2}}{(\varphi_n,\varepsilon_2)} & \dots & \frac{\varphi k_{n\,n-1}}{(\varphi_n,\varepsilon_{n-1})} & \frac{\varphi}{(\varphi_n,\varepsilon_n)} \end{array} \right\Vert$$

(c)

Figure 4: TeX4ht jsMath output for a LaTeX input.

the input, it places in the `dvi` file a request that the character will be typeset by the first symbol of the `cmsy` font. When TeX4ht encounters the request while processing the `dvi` file, it opens an alternative hypertext font file of its own, named `cmsy.htf`, and retrieves the first entry in that file. This entry gives the Unicode value `−`.

Given this Unicode value `−`, the TeX4ht utility searches for the value in the active encoding file `unicode.4hf`. If the value `−` is not found in the encoding file, it is inserted as is into the output. If the value is found in the encoding file, the replacement from the encoding file is instead placed in the output.

The flag '`-cmozhtf`' of the command line requests an encoding file that does not include an entry for `−`. Consequently, in the default setting, the Unicode value is placed in the output. The following steps show how the simple '`-`' character can be output instead:

- Make a copy of the following font encoding file,

  `ht-fonts/mozilla/charset/unicode.4hf`

- Add the following record to the new file:

  ```
  '&#x2212;' '' '-' ''
  ```

- Specify the location of the new encoding file on the command line. If the file is in the current directory, the command line can take the following form:

  ```
  htlatex file "xhtml,jsmath"
  ```

3.2 Preventing the expansion of symbol macros

A redefinition of the control sequences `\pm` and `\ne` in the following manner prevents the expansion of the symbol macros, respectively, into '`±`' and '`\mathrel{≠}`':

```
\edef\pm{\HCode{\string\pm\space}}
\edef\ne{\HCode{\string\ne\space}}
```

3.3 Transformations involving rewriting

A translation of the `\over` operator into the `\frac` function can be achieved by introducing the following TEX4ht configuration.

```
\Configure{over}
  {\Send{GROUP}{0}{\string\frac\l:brace}}
  {\HCode{\r:brace\l:brace}%
   \Send{EndGROUP}{0}{\r:brace}}
```

The configuration relies on capabilities to inversely process DVI code into source code. The next observations provide some insight into how the configuration works:

1. The arguments of a configuration

   ```
   \Configure{over} {...} {...}
   ```

   are inserted by TEX4ht immediately before and after the `\over` operator.
2. The `\Send{GROUP}{0}{...}` instruction sends its argument backward to the start of the current group.
3. The `\Send{EndGROUP}{0}{...}` code delivers its argument forward to the end of the current group.
4. The contribution of `\Configure{over}` to the code fragment `{...\over...}` provides the following initial outcome.

   ```
   {...
   \Send{GROUP}{0}{\string\frac\l:brace}
   \over
   \HCode{\r:brace\l:brace}
   \Send{EndGROUP}{0}{\r:brace}
   ...}
   ```

5. After applying the `\Send` instructions, the code takes the following form.

   ```
   {\frac\l:brace ...
   \over
   \r:brace\l:brace ... \r:brace}
   ```

6. The braces '`{`' and '`}`' and the `\over` operator do not introduce content to the output file. Consequently, the net contribution is as follows, where `\l:brace` and `\r:brace` produce left and right braces, respectively.

   ```
   \frac\l:brace ...\r:brace\l:brace
               ...\r:brace
   ```

4 Speech markup and synthesis

XML and Cascading Style Sheets (CSS) conventions are commonly and increasingly being used for describing the desirable rendering of documents into visual forms. Similar attention is also being given to the development of analogous standards for rendering documents in audio forms.

The aural conventions are concerned with properties such as pitches, volumes, rates, and pauses to be associated with the different parts of the documents. The conventions are in particular valuable for introducing annotations that highlight the structural characteristics of documents.

Of particular interest in this regard are the draft proposals from the W3C consortium of the Speech Synthesis Markup Language (SSML) [16] and Aural Cascading Style Sheets (ACSS) [1]. Figures 5(a) and 5(b), respectively, illustrate the notations in those proposals. The SSML specifications are based on, and are similar to, the Java Speech Markup Language (JSML) specifications [7].

Emacspeak [13] supports a restricted variant of ACSS, recognizing the properties of voice-family, stress, richness, pitch, and pitch-range. The C++ program of Figure 5(c) renders, on the Microsoft Windows Vista platform, files in SSML format. The Java program of Figure 5(d) renders JSML files, on platforms offering a JSML-based implementation to the Java Speech APIs [6].

5 From LaTeX to speech

Audio representations of documents are of great importance for people with print disabilities [14] as in many cases they have no alternative ways to access the content of the documents. Yet, documents in audio formats can be also useful for the general public. For instance, that might be the case for people who want to listen to a document while driving, or for

```
<speak> <p>Take a deep breath <break strength="weak"/> then
        <prosody rate="-10%">speak slower</prosody>.</p>
        <p>Also <prosody volume="loud">raise your voice</prosody>
        so everyone will hear you.</p>                     </speak>
```

(a)

```
h1 { voice-stress: strong; voice-rate:-10%; pause-after: 20ms; }
msqrt.before { content: "Square root: " }
msqrt.after  { content: "End root. " }
```

(b)

```
#include <sapi.h>
int main(int argc, char* argv[]){
  ISpVoice * synth = NULL;
  if (FAILED(::CoInitialize(NULL))){ return 0; }
  HRESULT hr = CoCreateInstance(CLSID_SpVoice, NULL,
                CLSCTX_ALL, IID_ISpVoice, (void **)&synth);
  if( SUCCEEDED( hr ) ){
    int n = strlen(argv[1]);
    wchar_t *s = (wchar_t *)  malloc(n+1); s[n] = '\0';
    while( n-- > 0 ){ s[n] = argv[1][n]; }
    hr = synth->Speak(s, SPF_IS_FILENAME | SPF_PARSE_SSML, NULL);
    synth->Release();
    synth = NULL;
  }
  ::CoUninitialize(); return 0;
}
```

(c)

```
import javax.speech.*;
import javax.speech.synthesis.*;
import java.net.*;
import java.io.*;
public class Speaker{
  public static void main(String args[]) {
    try {
      Synthesizer synth = Central.createSynthesizer(new SynthesizerModeDesc());
      synth.allocate();
      synth.resume();
      synth.speak(new File(args[0]).toURI().toURL(), null);
      synth.waitEngineState(Synthesizer.QUEUE_EMPTY);
      synth.deallocate();
    }catch( Exception e ){
      System.err.print("--- ERROR --- "); e.printStackTrace();
} } }
```

(d)

Figure 5: (a) SSML. (b) ACSS. (c) SSML file speaker. (d) JSML file speaker.

authors wishing to "proof listen" their writings in addition to (or instead of) proof reading.

LaTeX documents can be translated by TeX4ht into files annotated for speech [5]. For the ACSS speech variant of Emacspeak the requests can be made with commands similar to the following:

```
eslatex file
```

For output in JSML format the calling commands can be as follows:

```
jslatex file
```

TeX4ht configurations similar to those provided for the JSML and the Emacspeak variant of ACSS can, and in time will, be also tailored for output modes in SSML and the W3C version of ACSS.

It should be noted that currently no browser is available for effectively inspecting and navigating highly structural content in audio mode. In fact, it is even not clear what features audio browsers should offer to tackle this issue. Such a deficiency makes it very difficult to use audio resources to study technical topics, including those relying heavily on mathematical notations. In addition, it makes it difficult to decide what added information TEX4ht should provide in the translated material to enhance its accessibility.

Much of the approach for audio rendering of mathematics is motivated by Nemeth braille and expressed in MathSpeak [9]. The audio cues for different logical elements of data might also be specified within browsers instead of being provided to them within the data. MathPlayer [8], for instance, behaves so in rendering MathML expressions into audio. LATEX files can be transformed by TEX4ht to satisfy MathPlayer requirements with commands of the following form:

```
mzlatex file "xhtml,mathplayer"
```

The pioneering work of automatically putting technical content into audio format is due to T.V. Raman and assumed the LATEX language for the input data [11, 12].

Acknowledgement

I am grateful to Barbara Beeton, Karl Berry, and Susan Jolly for their valuable input.

References

[1] *Aural style sheets*, W3C Working Drafts, `http://www.w3.org/TR/CSS21/aural.html`, `http://www.w3.org/TR/css3-speech/`.

[2] D. Cervone, *jsMath: A Method of Including Mathematics in Web Pages*, `http://www.math.union.edu/~dpvc/jsMath/` and `http://sourceforge.net/projects/jsmath/`.

[3] *Duxbury Braille Translator (DBT)*, Duxbury Systems, `http://www.duxburysystems.com/`.

[4] E. Gurari, *TEX4ht*, `http://www.cse.ohio-state.edu/~gurari/TeX4ht`.

[5] E. Gurari, *LaSpeak: LATEX and Speech*, `http://www.cse.ohio-state.edu/~gurari/laspeak`.

[6] *Java Speech API*, Sun Microsystems, `http://java.sun.com/products/java-media/speech/` (version 1) and `http://jcp.org/en/jsr/detail?id=113` (version 2, proposed draft, 11 June 2007).

[7] *Java Speech Markup Language (JSML)*, Sun Microsystems, 1999, `http://java.sun.com/products/java-media/speech/forDevelopers/JSML/index.html`.

[8] *MathPlayer*, Design Science, `http://www.dessci.com/en/products/mathplayer/`.

[9] *MathSpeak Core Specification Grammar Rules*, `http://www.gh-mathspeak.com/examples/grammar-rules/`.

[10] *MediaWiki*, `http://www.mediawiki.org/wiki/MediaWiki`.

[11] T. V. Raman, *An audio view of (LA)TEX documents*, *TUGboat* 13:3, October 1992, 372–379, `http://www.tug.org/TUGboat/Articles/tb13-3/raman.pdf`.

[12] T. V. Raman, *Audio System for Technical Readings (AsTeR)*, Ph.D. Dissertation, Cornell University, May 1994, `http://emacspeak.sourceforge.net/raman/publications/web-aster/root-thesis.html`, Examples: `http://www.cs.cornell.edu/home/raman/aster/aster-toplevel.html`.

[13] T. V. Raman, *Emacspeak — The Complete Audio Desktop*, `http://emacspeak.sourceforge.net/`.

[14] Recording for the Blind and Dyslexic, `http://www.rfbd.org/`.

[15] *Scientific Notebook*, MacKichan Software, `http://www.mackichan.com/`.

[16] *Speech Synthesis Markup Language (SSML)*, W3C Working Draft, 11 June 2007, `http://www.w3.org/TR/speech-synthesis11/`.

[17] T. Wegrzanowski, *Texvc: TEX Validator and Converter*, `http://en.wikipedia.org/wiki/Texvc` and `http://meta.wikimedia.org/wiki/Help:Formula`.