

CERTAIN APPROACHES FOR A COMPLETE MATHEMATICAL TEXT-TO-SPEECH SYSTEM

S. GOMATHI @ ROHINI

Assistant Professor, Department of Computer Science,
Sri Ramakrishna Engineering College,
Vattamalaipalayam, NGGO Colony, Coimbatore 641022, India
rohinismohan@yahoo.com

R.S.UMADEVVI

II Year MCA, GRG School of Computer Technology,
Avanashi Road, Coimbatore 641 004, India
umapragaladh@gmail.com

Abstract

This paper discusses the teaching - learning process of mathematical subjects made accessible to visually disabled students. It aims to provide users a friendly vocal interface with computer and allow visually impaired people to use the computer. It focuses on the issues that arise from representation, encoding and manipulation of mathematics in electronic media.

Professionals use tools like LaTeX that result in encoding of mathematics, symbols and expressions. A well encoded MathML expression can be evaluated in a computer algebra system rendered in a web browser. Some approaches for achieving the transformations from MathML to speech output are mentioned in this paper. A Text-To-Speech (TTS) system that supports JSML is responsible for rendering a document as spoken output. The challenge for educational institutions is, as they increase the use of e learning, they must apply these techniques in a way that does not introduce unnecessary barriers to disabled learners.

Keywords: synthesiser, vocal interface, constructs.

1. Introduction

In recent years, the use of computers in speech recognition and synthesis has become an important area of study among computer scientists. They aim to provide a user-friendly vocal interface with computer and allow visually impaired people to use the computer. The recent developments in speech and computer technology have produced unrestricted vocabulary speech synthesis on personal computers in English and some other European languages.

2. Scope

Based on the survey, over 45 million people around the world are completely blind. 180 million more people are legally blind and seven million people are diagnosed as blind or legally blind every year. One of the greatest hurdles with blinds to enter careers in science, technology or mathematics is the paucity of tools to help them

read and write equations. Over the years, there have been numerous projects around the world with the goal of building special tools to help the visually impaired students read and write equations.

3. Mathematics in Print

3.1. *LaTeX*

LaTeX is a document preparation system for the TeX typesetting program. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including important features for mathematics. LaTeX has become the dominant method for using TeX. Some blind mathematicians have developed the skill to read LaTeX code directly. However this approach is not judged appropriate for most visually impaired students. Because it requires a high level of abstract mental ability to parse the code and extract the semantics from a mark-up of visual presentation of the mathematics. Many professional mathematicians, authors or editors of texts with mathematics use tools like LaTeX that result in encoding of mathematics, symbols and expressions.

3.2. *MathML*

MathML is an application of Extended Markup Language (XML) for describing mathematical notations and capturing their structure and content. It aims at integrating mathematical notations into World Wide Web (WWW) documents, so that they can be accessible to the visually impaired

The Mathematical Mark-up Language (MathML) was first introduced by the WWW Consortium (W3C) in 1998 with the intention of enabling mathematics to be served, received and processed on the web, just as HTML. The latest version is MathML Version 3.0, published in October 2010.

MathML enables the structure and parameters of mathematical expressions to be encoded. A well encoded MathML expression can be evaluated in a computer algebra system rendered in a web browser, edited in a word processor or printed. In MathML, there are two styles of encoding called content markup encodings (focusing on the underlying meaning of the expression) and presentation markup encodings (focusing on how the expression will be displayed or printed), which will be used depending on the purpose of the application. MathML allows users to use either kind of encoding or mix them as a hybrid.

The primary role of MathML content markup encoding is to provide a mechanism for recording a particular notational structure with its particular mathematical meaning. i.e., every content element must have a mathematical definition associated with it in some form. Content markup is intended for facilitating applications other than display, like computer algebra and speech synthesis. As a consequence, when using a content mark-up, it is harder to control how an expression will be displayed.

In presentation markup encoding, expressions are built-up using layout schemata, which tell how to arrange their sub-expressions, like a fraction or a superscript. The way the MathML layout schemata are nested together is naturally described by an expression tree, where each node represents a particular schema and its branches represent its sub-expressions. Even complicated, nested expressions are built-up from a handful of simple schemata.

Content mark-up and presentation mark-up use the same kind of syntax. Each layout schemata or content construction corresponds to a pair of start and end tags. The mark-up for each sub-expressions is enclosed between the start and end tags; the order they appear determines what roles they play in the given schema. MathML presentation mark-up for the following equation (Eq.1) is given below as example.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (1)$$

```

<mrow>
<mi>x</mi>
<mo>=</mo>
<mfrac>
<mrow>
<mrow>
<mo>-</mo>
<mi>b</mi>
</mrow>
<mo>&PlusMinus;</mo>
<msqrt>
<mrow>
<msup>
<mi>b</mi>
<mn>2</mn>
</msup>
<mo>-</mo>
<mrow>
<mn>4</mn>
<mo>&InvisibleTimes;</mo>
<mi>a</mi>
<mo>&InvisibleTimes;</mo>
<mi>c</mi>
</mrow>
</mrow>
</msqrt>
</mrow>
<mrow>
<mn>2</mn>
<mo>&InvisibleTimes;</mo>
<mi>a</mi>
</mrow>
</mfrac>
</mrow>

```

3.3. Web browsers supporting MathML

The W3C's MathML group maintains a list of MathML related software that includes browsers and browser plug-ins. However, even if a particular browser supports MathML, it also supports all accessibility potential anticipated in adopting MathML as the encoding scheme. MathML can also be embedded in Word documents and equation editors.

- Mozilla Version 1.1 and higher: all platforms (e.g.Windows, Mac, Linux/Unix)
- Netscape Version 7.1 : all platforms
- Netscape Version 7.0 : all platforms except Mac
- Internet Explorer Version 6.0 : with MathPlayer
- Plug-in : Windows only

3.4. MathML to speech approaches

A mathematics professor's written form of lecture is needed to remove ambiguities from that of spoken form. That being said, speech is an approach that has some benefit for those learning to access mathematics in electronic media. Transforming text to speech output still suffers from the challenge of how to remove ambiguity. Some approaches for achieving the transformations from MathML to speech output are mentioned here.

3.4.1. *MathML to text to voice*

If MathML in a web page is transformed to text, this text can be transformed to voice, if it is passed to a screen-reader, as other text is read in the Web page.

3.4.2. *MathML to voice mark-up languages*

The Java Speech API Markup Language (JSML) is a text format, used by applications to annotate text input to speech synthesizers. JSML elements provide a speech synthesizer with detailed information on how to speak text and thus enable improvements in the quality, naturalness and comprehension of synthesized speech output. JSML defines elements that describe the structure of a document, provides pronunciations of words and phrases, indicates phrasing, emphasises pitch and speaking rate and controls other important speech characteristics.

3.4.3. *TEX/LaTeX to hypertext documents*

TeX4t is a system that converts TEX/LaTeX inputs into various hypertext documents (HTML, XML or MathML). The TEX4ht translates general LaTeX constructs into the restricted dialects recognizable by TTS engines. The jsMath dialect for rendering LaTeX through JavaScript is employed. AsTeR program automatically renders LaTeX documents into audio. Newer audio browsers are expected to address XML documents that adhere to the JSML and ACCS specifications.

TEX4ht translates LaTeX to XML-based representations that support speech, which has been implemented and tested successfully on Fedora Core 3 platform. The results have been evaluated to determine the performance of the synthesizer developed in terms of intelligibility and naturalness.

Adaptive Technology Resource Centre (ATRC) at the University of Toronto has explored the potential in transforming from MathML mark-up to the appropriate format for input into a speech synthesizer. It has further explored approaches for using the advanced features of JSML to study how the synthesiser pronounces the mathematics expressions. This group has also used voice style sheets to define sets of these tailoring appropriate to different contexts like working with algebra, calculus or vectors etc. This can be a significant asset in removing the potential ambiguities of working with MathML presentation mark-up as opposed to content mark-up. Many other research centers work on similar approaches based on VoiceXML or ActiveX (Dynamic Link Library) DLLs to a Text-To-Speech (TTS) engine. However, none of these approaches can be said widely adopted by disabled people.

4. Steps in Speech Synthesis Process

A Text-To-Speech (TTS) system that supports JSML is responsible for rendering a document as spoken output and using the information contained in the markup to render the document as intended by the user.

Document Creation: A text document produced automatically, by human authoring or combination of these forms is provided as input to the synthesis processor. JSML defines the form of the document.

Document Processing: The following six steps are undertaken by a synthesis processor to convert markup text input into automatically generated voice output. The processor has the ultimate authority to ensure what it produces is pronounceable. In general, the markup provides a way to make prosodic and other information

available to the processor, which would be unable to acquire on its own. Then the processor determines whether and in what way to use the information.

4.1. *XML parsing*: An XML parser is used to extract the document tree and content from the incoming text document. The structure, tags and attributes obtained in this step influence each of the following steps.

4.2. *Structure analysis*: The structure of the document influences the way in which a document should be read. For example, there are common speaking patterns associated with paragraphs and sentences.

In English, "£30" may be spoken as "thirty pounds". Similarly, "1/4" may be spoken as "quarter", "January four", "April first", "one of four" and so on.

4.3. *Text normalization*: All written languages have special constructs that require a conversion of the written form into the spoken form. Text normalisation is an automated process of the synthesis processor that performs this conversion.

By the end of this step, the text to be spoken will be converted completely into tokens. The exact details of what constitutes a token are language-specific. In English, tokens are words, separated by white space.

4.4. *Text-to-phoneme conversion*: Once the synthesis processor has determined the set of tokens to be spoken, it must derive pronunciations for each token. Pronunciations may be conveniently described as sequences of phonemes, which are units of sound in a language that serve to distinguish one word from another. Each language has a specific phoneme set. US English has around 45 phonemes and Chinese has 422. This conversion is made complex by a number of issues.

One issue is that there are differences between written and spoken forms of a language, and these differences can lead to indeterminacy or ambiguity in the pronunciation of written words. In many languages the same written word may have many spoken forms. For example, in English, "caught" may be spoken as "cot". Both human speakers and synthesis processors can pronounce these words correctly in context, but may have difficulty without context.

Another issue is the handling of words with non-standard spellings or pronunciations. For example, an English synthesis processor will often have trouble determining how to speak some non-English-origin names, e.g. "catamaran" (pronounced "katamaran").

4.5. *Prosody analysis*: Prosody is the set of features of speech output that includes the pitch, the timing, the pausing, the speaking rate, the emphasis on words and many other features. Producing human-like prosody is important for making speech sound natural and for correctly conveying the meaning of spoken language. JSML provides the content to be spoken, logical descriptions of style, the break and prosody elements and thus coexists with uses of the emphasis element and the processor's own determinations of prosodic behavior.

4.6. *Waveform production*: The phonemes and prosodic information are used by the synthesis processor in the production of the audio waveform. There are many approaches for this step, so there may be considerable processor-specific variation.

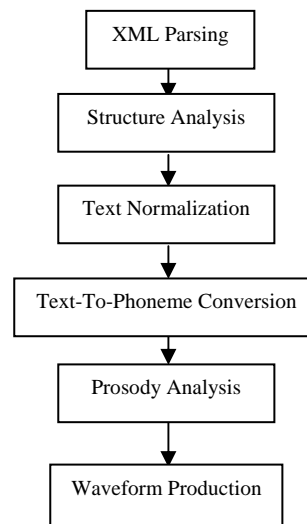


Figure 1. Speech Synthesis Process

5. Future Work

Incremental advances are anticipated over the next few years on MathML based approaches based on improved browser support, editors and even extensions to the language, but other approaches may supersede this. There are various research teams across the world working on the challenges of encoding mathematics in a way that is truly accessible.

This effort pays a deeper look at the way people interact with symbolic languages in general. It maintains past efforts that have only limited success as they have concentrated on the transformation of visual presentation and have not considered the deeper issues of the cognitive apprehension and manipulation of the symbolic languages. A key tactic in the proposed solution is a total separation of semantics from the presentation. This has been partially achieved with MathML.



6. Conclusion

This paper has concentrated on the challenges of encoding mathematics in web pages and electronic documents, so that disabled students can access them in their studies. However, it must be stated that despite of these challenges, many disabled students successfully study mathematics based subjects by using these tools. The challenge for educational institutions is, as they increase the use of e learning, they must apply these techniques in a way that does not introduce unnecessary barriers to disabled learners. Although the current state-of-the-art is less than ideal, approaches are available to make mathematics in electronic documents and on the web accessible to virtually all disabled students. Furthermore, professors should learn to recognise the disabled students' own expertise in developing coping strategies that would work for them.

References:

1. www.unicode.org
2. www.ctan.org
3. www.nbd.org
4. www.w3.org/Math/
5. www.swiss.csail.mit.edu/projects/intelligent-book/
6. www.mathml.com
7. www.latex-project.org
8. web.mit.edu/ist/topics/webpublishing/

9. www.dessci.com/en/products/mathplayer/
10. www.java.sun.com/products/java-media/speech/forDevelopers/JSML/
11. www.section508.gov
12. <http://www.cs.berkeley.edu/~fatemam/papers/voice+hand.pdf>
13. <http://www.sny.jussieu.fr/inova/villette2002/act5b.htm>
14. http://www.csun.edu/~hcmth008/mathml/acc_tutorial.pdf
15. <http://people.cs.vt.edu/blaha/docs/Latex%20reference.pdf>
16. <http://www.citeulike.org/group/15030/article/460173>
17. http://www.csun.edu/~hcmth008/mathml/converting_to_mathml.pdf
18. http://www.csun.edu/~hcmth008/csun_conference/csun_conference.pdf
19. <http://www.w3.org/Math/Software/>
- 20.

	<p>S. Gomathi @ Rohini is a M.Sc., M.Phil. graduate from St. Xavier's College, Palayamkottai with specialisation in computer science. She owns more than 6 years of teaching experience and 3 years of industrial experience.</p> <p>Currently she is doing her research in computer science at Mother Teresa Womens' University, Kodaikanal. She has undergone training in various domains. She has published papers in many national conferences.</p> <p>Her areas of interest are mainframes, computer networks and NLP. Now she serves as Assistant Professor in Sri Ramakrishna Engineering College, Coimbatore.</p>
	<p>R.S. Umadevi is an MCA student from G.R.G. School of Applied Computer Technology, Coimbatore. She has a rich blend of analytical and technical skills. Her areas of interest are image processing, operational research and microprocessor.</p>

Running head: vocal interface with computer to allow visually impaired

Appendix:

MathML presentation examples

1. If no trimming attributes are specified, then the complete document is rendered:

```
<math xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis11/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <mark name="mark1"/>
  <audio src="middle.wav"/>
  <mark name="mark2"/>
```

```
<audio src="last.wav"/>
</speak>
```

Here "first.wav", "middle.wav" and "last.wav" are rendered, and "mark2" is the last mark rendered.

2. The startmark can be used to specify the rendering begins from a specific mark:

```
<speak startmark="mark1" version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis11/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <mark name="mark1"/>
  <audio src="middle.wav"/>
  <mark name="mark2"/>
  <audio src="last.wav"/>
</speak>
```

Here "middle.wav" and "last.wav" are rendered, but not the "first.wav", since it occurs before the startmark "mark1".

3. The end of rendering can be specified using the endmark:

```
<speak endmark="mark2" version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis11/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <mark name="mark1"/>
  <audio src="middle.wav"/>
  <mark name="mark2"/>
  <audio src="last.wav"/>
</speak>
```

Here "first.wav" and "middle.wav" are completely rendered but none of "last.wav" is rendered.

4. The trimming attributes can be used to control both the 'start' and 'end' of rendering:

```
<speak startmark="mark1" endmark="mark2"
  version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis11/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <mark name="mark1"/>
  <audio src="middle.wav"/>
  <mark name="mark2"/>
  <audio src="last.wav"/>
</speak>
```

Here only "middle.wav" is played.