Supporting Visual Impaired Learners in Editing Mathematics

Volker Sorge
Progressive Accessibility
Solutions, Ltd.
University of Birmingham, UK
V.Sorge@progressiveaccess.com

ABSTRACT

We present an extension to the Emacspeak audio desktop that provides support for editing mathematics in LaTeX. It integrates MathJax and the Speech Rule Engine to support users in reading LaTeX sources, manipulating mathematical formulas and verifying correctness of the rendered output.

Keywords

STEM Accessibility, Mathematics, MathJax

1. INTRODUCTION

Mathematics is still the single most significant hurdle for inclusive education for visually impaired students in the STEM subjects. In particular, in secondary education the increasing complexity of mathematical formulas makes it harder to communicate content equally effectively to both sighted and blind students. While mathematical formulas can be expressed in dedicated Braille formats, these become quickly unwieldy for more advanced mathematics and moreover it becomes impossible for students to communicate them to their peers or professors. Consequently they need to start using a linear format that is commonly understood. And indeed on transitioning from high school to University students often have to learn LaTeX to have a means to read and communicate advanced material.

While there are some specialist editors that support editing mathematics for the blind (e.g. [2]) there is very little support for reading or authoring LaTeX documents directly (cf. [3]). In particular, it is difficult for students to write mathematical content in LaTeX while checking that the rendered output indeed corresponds to the mathematical formula they intended to write. We aim to support this task by extending the Emacspeak audio desktop [4] with a mathematics option. It exploits the power of MathJax [1] to translate LaTeX expressions into MathML markup and the Speech Rule Engine (SRE) [5] to translate MathML into speech strings that can be passed to a TTS.

ASSETS '16 October 23-26, 2016, Reno, NV, USA

ACM ISBN 978-1-4503-4124-0/16/10.

DOI: http://dx.doi.org/10.1145/2982142.2982212

Our work focuses on supporting students in the aspects of learning, writing and working with LaTeX. In particular, to allow them to take source material, read it, browse it and manipulate it. We support learning LaTeX by offering ways to write and rearrange expressions and hear the effect, by using MathJax's error reporting mechanism to indicate incorrect expressions and by enabling interactive exploration of rendered expressions on the fly, while editing. As editing mathematics is already a time consuming task we aim to minimise the impact of having to listen to lengthy expressions multiple times by exploiting Emacspeak's prosody feature and SRE's ability to summarise sub-expressions meaningfully for concise aural rendering of expressions.

2. TECHNICAL REALISATION

Technically we realise our aims by combining Emacspeak with MathJax and the Speech Rule Engine.

Emacspeak is a speech interface that allows visually impaired users to employ the Emacs editor as the main tool for desktop activities. It supports tasks from implementing in different programming languages, writing documents in markup formats like markdown or LaTeX, to browsing the file system and the WWW. Emacspeak uses audio formatting via W3C's Aural CSS (ACSS) to produce rich aural presentations of electronic information with the aim of shifting some of the burden of listening from the cognitive to the perceptual domain. For instance, syntax highlighting is aurally rendered by changes in voice characteristic and inflection. Combined with appropriate use of non-speech auditory icons this creates the equivalent of spatial layout, fonts, and graphical icons important in a visual interface.

Speech Rule Engine (SRE) is a Javascript library that translates XML expressions into speech strings according to rules that can be specified in an extended Xpath syntax. It was originally designed for translation of MathML and MathJax DOM elements for Google's ChromeVox screen reader [6]. It has since been turned into a stand-alone application that can be installed server side as a NodeJS application as well as used in client side web applications. Besides the rules originally designed for the use in ChromeVox, it also has an implemententation of the full set of Mathspeak rules. In addition it contains a library for semantic interpretation and enrichment of MathML expressions, rules for intelligent summarisation of sub-expressions, facilities for the interactive exploration of mathematical expressions, highlighting of DOM nodes, etc. SRE runs both in browsers as well as in NodeJS. It can be installed via Node's package

manager npm and offers a command line interface for batch translation of XML expressions.

MathJax is a JavaScript library for visual rendering of formulas across all platforms and browsers. It was originally built as a polyfill solution for rendering mathematics in browsers until the MathML standard would be sufficiently supported. But since most browsers still do not support MathML natively, MathJax has become the quasi standard for displaying Mathematics on the web. MathJax can render the most common mathematical authoring formats —- LaTeX, ASCIIMath, and MathML — into high quality output represented either as HTML or as SVG. Although MathJax is built as a client side solution, to be included into webpages, there is a special NodeJS package release, which allows server side rendering of mathematical expressions from one of the three input formats into SVG. In addition, it offers a way to convert LaTeX into standardised MathML. MathJax can also use SRE as a dependency to produce speech strings for mathematical expression for inclusion into the rendered output.

NodeJS Bridge: The three systems are combined with a simple NodeJS bridge, whereby MathJax and SRE run within an inferior JavaScript process in Emacs. Emacspeak communicates with both systems by message passing between the different Emacs processes. Using an inferior process over a simple batch process integration does not only decrease potential lag due to delay in system startup but also has the advantage that both integrated systems can keep an independent internal state Emacspeak can refer to.

3. READING LaTeX IN EMACSPEAK

LaTeX formulas are spoken in Emacspeak in two different ways: Either integrated into the continuous flow when the entire document is read or speaking can be triggered interactively for a single formula. In either case Emacspeak detects the math delimiters and sends the expression to the node process. MathJax then translates it into MathML and passes it to SRE, which in turn performs a semantic analysis of the expression and then translates the expression into a speech string using a chosen set of speech rules.

One of Emacspeak's features is that it enables audio formatting, by employing changes in prosody (e.g., changes in pitch or stress), to indicate syntax highlighting, special text elements like links etc. Similarly, SRE supports prosody markup by allowing speech rules to define changes in pitch, rate, volume and by adding pauses. We combine this by translating mathematical expressions in SRE using a speech rule set that indicates two dimensional layout, such as sub and superscripts or positions in fractions with changes in pitch and rate, rather then employing more verbose indicator words like StartFraction or EndFraction. SRE then returns translated formulas as sexpressions that contain speech strings together with Aural CSS markup that Emacspeak can exploit for audio formatting.

Since mathematical formulas are often of a complex nature and just listening to an expression once is generally not enough for a user to fully take in its meaning, it is particularly important to be able to engage with them interactively by offering a means of traversing sub-expressions. SRE provides a number o different navigation options for mathematical expressions, which we exploit from Emacspeak. Thereby SRE keeps an internal state on the latest mathematical formula that it has rendered. Emacspeak then allows users to walk this formula using simple cursor key navigation, while SRE produces speech strings for the corresponding sub-expressions, potentially together with positional information, such as denominator, numerator etc.

4. LEARNING SUPPORT

One major difficulty presented to students new to LaTeX is learning the intricacy of its syntax and in particular to correctly implement complex, nested mathematical expressions. We provide a number of features that are aimed in particular at helping students understand expressions and write correct ones themselves. For this we exploit as much as possible the abilities of the systems we integrate:

Error Handling Since MathJax implements a version of the TeX typesetting system it can return fairly detailed error reports. We expose these directly to the reader together with an error earcon to allow users to quickly understand and rectify problems in their input expressions.

Syntax Highlighting Since LaTeX documents are implemented rather than composed in a WYSYG style, editors help authors to distinguish content from markup using syntax highlighting. Emacspeak can exploit its audio formatting to achieve a similar effect, making it possible to separate content from markup and giving the reader an indication of what layout elements markup would produce, such as bold type face, section headings etc. For math expressions audio formatting is used to indicate nesting depth of braces, giving a user an indication of the position within an expression they edit thus highlighting potential problems or errors.

Minimise Aural Rendering Since editing LaTeX can be a time consuming task, which is only compounded by the need to listen to lengthy audio output of mathematics, one of our main goals is to decrease the burden of listening. One way is by prefering rich audio formatting to more verbose traditional reading styles of mathematics (cf. MathSpeak for example). A second way is by exploiting the summarisation features SRE provides. For example, instead of reading a complete expression, SRE can use its semantic interpretation to produce summaries such as "sum with 5 summands", which are only expanded when explicitly prompted by a user, thus shortening aural output while editing.

Acknowledgements

The author is grateful for TV Raman's technical work on the Emacspeak integration.

5. REFERENCES

[1] MathJax Consortium. MathJax v2.6, 2015. http://www.mathjax.org.

[2] S. Dooley, D. Brown, E. Lozano, S. Park, and S. Osterhaus. Online nemeth braille input/output using content mathml. In *Proc of W4A*. ACM, 2016.

[3] NCBYS. Blindmath gems LaTeX. http://www.blindscience.org/blindmath-gems-latex.

[4] TV Raman. Emacspeak. http://emacspeak.sourceforge.net.

[5] V. Sorge. Speech Rule Engine. https://github.com/zorkow/speech-rule-engine.

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