2016
THE JOURNAL ON TECHNOLOGY AND PERSONS WITH DISABILITIES

Generating Content MathML Markup from Nemeth Braille Input Sequences

Samuel S. Dooley, Dan Brown, Edgar Lozano
Pearson Assessments
sam.dooley@pearson.com, dan.brown@pearson.com, edgar.lozano@pearson.com

Abstract

The need for high-quality braille materials is one of the most challenging obstacles faced by visually-impaired students, especially for mathematical content. The need for visually-impaired students to communicate their work to sighted instructors often prevents them from participating in mainstream classrooms in STEM subjects. Backward translation from Nemeth Braille to print mathematics is a difficult and time-consuming process even for teachers of the visually impaired, who may not have an extensive understanding of the mathematical concepts underlying the notation. The goal of this current work is to provide a software system that supports the automatic backward translation of Nemeth Braille input, in the context of a WYSIWYG equation editor designed for sighted math users. This software allows a Nemeth Braille user who is unable to access the printed form of an equation to produce high-quality print mathematical formulas in a fraction of the time, and at a fraction of the expense, of current best practices for braille-to-print translation. This software provides the basis for more immediate communication of mathematical content from visually-impaired Nemeth Braille users to sighted instructors and peers, reducing communication barriers that inhibit visually-impaired students from participating in mainstream math and science classrooms.

Keywords

Mathematical User Interfaces; Equation Editors; Nemeth Braille; Content MathML; Braille Translation Software.

Introduction

The W3C Math Markup Language has now been used for over fifteen years to represent structural forms (Content MathML) and print notations (Presentation MathML) for online mathematical expressions (Ion and Miner). Nemeth Braille has now been used for over sixty years as a means to capture print math notation for tactile reading for low-vision and blind users (The Nemeth Braille Code for Mathematics and Science Notation). While Nemeth Braille has much in common with Presentation MathML in its intent to capture the appearance of print math notation, the design of its coding rules shares in common with Content MathML its desire to preserve the structural form of the expressions it encodes.

Content MathML provides a common target representation that can be created by keyboard input events using a key event-based transformation rule framework to effect the construction of mathematical expressions (Dooley, 55-62). Using this framework, input key event sequences from a braille keyboard may be used to invoke input transformation rules to create, delete, and/or modify a current expression in a manner that simulates Nemeth Braille mathematical expression entry, but which creates Content MathML expressions as the output of the keying process. In this fashion, a common framework may be used to create Content MathML using a QWERTY keyboard for sighted users, and using a braille keyboard for visually-impaired users.

This framework enables the input from a sighted user to be displayed in a format that can be read by a visually-impaired user, and vice versa, where the simultaneous print and braille displays are updated upon receipt of each keystroke event, regardless of the type of keyboard used to produce the event.

Discussion

As described in Dooley, key-based transformations that modify one Content MathML expression to create a new Content MathML expression may be used to allow a QWERTY keyboard to create mathematical structures. These rules allow an equation editor to create, on each keystroke, the Content MathML markup that corresponds to the user's input. The Content MathML can then be transformed into Presentation MathML for print users, whether for online or offline use, and can be transformed into Nemeth Braille for visually-impaired users (Dooley and Park), who may access the braille output by means of refreshable braille devices or other tactile printing methods.

The input events generated by a braille terminal typically encode six- or eight-dot braille cells in ASCII braille, which allows each braille dot pattern to be treated as if it were a single character input event. These input events are transmitted from the hardware, via the user's screen reader software, to the equation editor. Then the same key-based transformation framework used to interpret key events from a QWERTY keyboard can be used to interpret braille events from a braille terminal.

For a large majority of the input rules for a QWERTY keyboard, one key event is enough to effect an immediate transformation in the current expression. In a similar fashion, a single braille cell that represents a specific symbol can be immediately encoded to create that symbol in the equation editor. Other Nemeth Braille encoding rules consist of sequences of braille cells that cannot be distinguished from each other by a single initial cell. As a result, the transformation rule framework for expression entry is augmented by a finite-state machine to track the state of the current input sequence while waiting until the end of the sequence to effect a transformation in the current expression. The finite state machine uses named input states to describe input sequences. As a simple example, a QWERTY keyboard produces a capital "A" using a single keyboard event, while a braille keyboard requires two cells (dot-6 dot-1) before the capital A can be added to the expression.

The finite-state machine is sufficient to encode rules that support the use of the numeric indicator (dots-3456) for the input of numbers, while allowing the numeric indicator to be omitted in situations where the following numerals are unambiguous. The baseline indicator is supported for moving outside the scope of subscripts and superscripts, which is more natural concept for a braille user than the use of arrow keys for a visual user. Type form indicators for

Greek upper-case and lower-case letters (including alternate forms), and script and Fraktur upper-case and lower-case letters are also implemented as special input states invoked by the natural braille cell sequences used to encode these characters.

More complicated examples of the use of the finite-state machine involve input sequences for shape indicators, negated operators, and composed relations. In some cases, one valid input sequence may be contained as a prefix to a longer input sequence for a different operator. For example, the input sequence for the less-than operator (dot-5 dots-13) may be extended by the horizontal bar character (dots-156) to form the less-than-or-equal operator. In these cases, the equation editor effects the input rule to create the less-than operator after receiving the first two cells, then replaces it with the less-than-or equal-operator after receiving the horizontal bar.

The input-rule transformation framework to create Content MathML from Nemeth Braille input has been implemented as a special browser test page that embeds the equation editor (Dooly), which receives key events from the QWERTY keyboard for visual users, and braille cell events encoded as ASCII braille from a refreshable braille device for visually-impaired users. The braille cells are received from the braille device via screen reader device drivers that deliver the input events to the equation editor embedded as a JavaScript component in a browser. Through this arrangement, a visually-impaired user can input math on the braille keyboard, and a sighted user can input math on the QWERTY keyboard, and both input streams can be directed to the same instance of the equation editor. A special input mode switch is used to communicate to the equation editor which keyboard is being used so it can apply the appropriate input transformation rules.

Conclusions

As described in Dooley, key-based transformations from named key events into Content MathML are well understood, and represent a specific case of a more general approach to the separation of application structure from user interface behaviors. These transformations allow math content forms to be created from many kinds of input events, including QWERTY key events as is done in other equation editors, and Nemeth Braille input cells, as described here.

Further work on the operator-based transformation framework and the fidelity of the Nemeth Braille input encoding it accepts would include addressing a few remaining issues in the

implementation of certain contextual input rules for certain math operators, completing the list of known operators to include limits, derivatives, and integrals, and extending the framework to support markup for content that includes both literary and mathematical expressions.

As time goes on, more and more of the means and methods by which math and science are taught are being transformed into electronic and/or online forms. Online math instruction and assessment are now high-profile lines of business for major publishing companies and consortia, and are transforming education at all levels. Electronic textbooks and other materials are increasingly replacing paper-based alternatives, and teacher-student interactions are taking place using social media, online meeting rooms, and distance learning systems. As these transformations take place with increasing speed, the accessibility of these solutions lags behind due to the inherent difficulty of providing accessible math software in an online world. Online math software presents unique challenges that are not found in text-based software, further compounding the difficulty in making such software accessible. Moving forward, the need for access to online math content will override all other concerns related to the communication of mathematics. If visually-impaired students are to succeed in mathematics in the classroom at all grade levels, their participation needs to be fully online and fully interactive. By reducing the time and cost involved in braille translation, this work has the potential to produce a truly level playing field for visually-impaired students in mainstream STEM classrooms.

Works Cited

Dooley, S. S. Editing Mathematical Content and Presentation Markup in Interactive Mathematical Documents, in Mora, T., Ed., Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, New York, 07--10 July 2002, ACM SIGSAM, ACM Press, pp. 55--62. Lille, France.

Dooley, S. S. and Park, S. Generating Nemeth Braille Output Sequences from Content MathML Markup, submitted to CSUN 2016.

Ion, P., and Miner, R., Eds. Mathematical Markup Language (MathML) 1.0 Specification, W3C Recommendation, W3C, 07 April 1998. http://www.w3.org/TR/1998/REC-MathML-19980407/

The Nemeth Braille Code for Mathematics and Science Notation, American Printing House for the Blind, Louisville, Kentucky, 1972. https://nfb.org/images/nfb/documents/pdf/nemeth_1972.pdf

Journal on Technology and Persons with Disabilities

ISSN 2330-4216

LIBRARY OF CONGRESS * U.S. ISSN CENTER
ISSN Publisher Liaison Section
Library of Congress
101 Independence Avenue SE
Washington, DC 20540-4284
(202) 707-6452 (voice); (202) 707-6333 (fax)
issn@loc.gov (email); www.loc.gov/issn (web page)