Building a Programmable Architecture for Non-visual Navigation of Mathematics: Using Rules for Guiding Presentation and Switching between Modalities

Iyad Abu Doush and Enrico Pontelli

Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, USA
{idoush, epontell}@cs.nmsu.edu

Abstract. This paper presents a new implementation framework for exploring different non-visual modalities of presentation and navigation of mathematical content. The objective is to create a framework to facilitate the investigation of different presentation modalities, and to sustain the development of solutions that could fit a broad range of visual disabilities. The increased flexibility and the ability to customize the presentation scheme have the potential to provide a better fit for the needs of the user and enhance accessibility of complex mathematical content. The proposed approach relies on two principles. The first principle is a separation of concerns: instead of freezing the modality of presentation in the navigation system, we introduce the notion of *rendering rule*. The second feature of the proposed navigation system is the ability to switch on demand between interactive navigation and prosody-based presentation. This work is implemented as an extension to the Firefox web browser. New functionalities for reading MathML contents in a web page are added to the open source screen reader FireVox.

Keywords: Rule-based Systems, Math Accessibility, Prosody, Navigation.

1 Introduction

The study of mathematics is crucial in the preparation of students to enter careers in science, technology, engineering and other disciplines, such as the social and behavioral sciences. For many sighted students, math education poses a serious roadblock in entering technical disciplines. The advent of the Internet has significantly enhanced accessibility of technical content, by making millions of documents available on-line and effectively breaking down a number of cultural and socio-economical barriers (e.g. reaching out to rural and under-represented regions). Unfortunately, this move has further deepened the divide between individuals with visually disabilities and those without. For the visually impaired students, the roadblock is now even higher [1].

Mathematical content represents a particular challenge for non-visual access, in most of its aspects – e.g., formulas, graphs, mathematical symbols, and abbreviated function names. Traditional communication channels employed by individuals with visual disabilities are inherently linear (e.g., audio, Braille). The 2-dimensional spatial nature of mathematical expressions and other mathematical contents (e.g., charts) is particularly hard to linearize in a manner that maintains an adequate level of information and without the introduction of an excessive overhead [2].

The literature is rich with proposals to address the problems of non-visual presentation of mathematical content. In particular, in recent years we have witnessed an increased interest towards *aural rendering* of mathematical content (mostly in terms of the presentation of mathematical formulae). The mainstream research focused on two aspects when presenting mathematical contents: *structural information and prosody indicators* [2].

Structural indicators rely on the introduction of specific aural items (typically speech components) to identify the relevant structural elements of the expression being read (e.g., boundaries of a fraction, application of a square root). Although providing structural information when presenting mathematical content is important to identify specific terms, a study by Stevens et al. [3] pointed to the problem of excessive load imposed by structural indicators on the reader's working memory.

The use of prosodic indicators can help visually impaired users in comprehending the mathematical contents, by relying on pauses and variations in tone and pitch. The importance of speech prosody for mathematical content has been highlighted by Fitzpatrick [4, 5]. This research emphasized that the use of speech rate and pauses can help in understanding the nesting levels in mathematical expressions. Nevertheless, pure prosody is limited when dealing with expressions with large sub-expressions.

The majority of the modern approaches [6, 7] for navigating and accessing mathematical contents render mathematical content in one way, by linearly speaking the mathematical expression from left to right – we will refer to this approach as *passive* presentation. More sophisticated approaches (e.g., [8]) use the expression tree as a way for navigating the mathematical expression in a hierarchical way – we will refer to this approach as *active* presentation. Several studies [9, 10] pointed also to the importance of analyzing the complexity of mathematical expressions to determine how to guide rendering, these rules have not been applied in concrete systems yet.

In this paper we propose a novel platform for experimenting with alternative methodologies for non-visual presentation of mathematical content. In our system, we will take advantage of both passive and active presentation approaches, and the user will be able to switch between the two schemes when navigating a mathematical expression. In the system we will introduce the explicit use of rendering rules to present the mathematical expression according to its complexity. Each rendering rule describes how a particular type of mathematical operator should be aurally presented; the rules are contextual, in that they can vary the presentation style depending on the context the operator appears in. The base of rendering rules is customizable, offering user modeling capabilities. The implementation of the system has been realized by extending FireVox [11], a Firefox extension used as a web browser screen reader, to read mathematical content encoded in presentation MathML format. We decided to extend FireVox since it is a cross platform that can read Firefox web browser content in different platforms (Windows, Linux, and Mac). It also imposes minimal installation requirements on the user. The extension allows the user to navigate the mathematical expression in the two different modalities. Providing the user with two modalities will help him/her in comprehending the mathematical expression.

2 Background

Passive versus Active Reading: Introducing more than one modality of presentation to the user has been shown to enhance comprehension of mathematical content. Raman [12] pointed to the problem of passive listener and suggested offering the user multiple views which will allow him/her to interact with the navigated content. Reddy [13] suggested providing the visually impaired user with the hierarchical structure of the equation, and allow the user to backtrack on specific parts of the equation.

Ferreira and Freitas [6, 7] developed a math expressions accessibility tool for reading MathML content, called AudioMath. The tool provides the user with only one navigation modality, which reads mathematical formulae from left to right, term by term, with possibility of backtracking on previous elements of the formulae.

In another work active reading for mathematical expressions is introduced. A plug-in for Internet explorer, called MathPlayer [14] allows the user to navigate mathematical expression in two schemes – text-based navigation and tree-based navigation. Another system for reading equations is MathGenie [15] which allows the user to navigate the mathematical equations using several schemes (left to right reading, move from term to term backward and forward, and equation structure overview). Gaura [8] presented REMathEx in which the user can navigate the presentation MathML as a tree structure. To show the grouping between sub-expressions a delay is used when the expression is aurally rendered.

Prosody versus Lexical Cues: Fitzpatrick [4, 5] proposed a methodology for presenting mathematical content by providing audio cues (loudness, duration, pitch, and pausing). This enhancement of the speech helps the user in comprehending the mathematical content. Fitzpatrick also suggested the use of speech prosody with the lexical indicators to understand complex mathematical expressions. Raman [12] confirmed that the use of pauses between sub-expressions can help the user by emphasizing the grouping of the content. He suggests the use of speech rate to denote nesting, pauses to denote grouping of sub-expressions, and pitch rate to reveal superscripts and subscripts.

Inserting structural information within spoken mathematical expression will help the user in grasping the expression much easier. For example, adding the lexical indicator "begin fraction" and "end fraction" can help the user in catching the division operation. Stevens et al. [3] pointed to the problem the user will have if the spoken text contains several lexical cues as the user hearing memory is limited. In order to improve the audio rendering of complex expressions Fitzpatrick [5] suggested the combination use of prosody and lexical indicators.

Cognitive Aspects: In order to present mathematical contents to the visually impaired in a normal way several researches conducted studies on the cognitive thinking when reading mathematical equations. Barraza et al. [16] showed that the equation readers looks abstractly at the mathematical equation and then starts understanding in depth the expression. The reader usually goes back again over the equation to understand or solve it. MathGenie equation reader applies cognitive psychology research conducted by Gillan et al. [15] for reading and comprehending mathematical equations.

Navigating the mathematical expression according to the complexity of the expression will help visually impaired users in understanding more complex expressions. Awde et al. [10] presented a framework for calculating the complexity of mathematical expressions rendered in the presentation MathML. The resulting complexity value is used to determine how to aurally present the mathematical content. Karshmer et al. [9] proposed a set of tools to help access mathematical contents encoded in LaTeX. They also mentioned that reading mathematical equations requires analysis of their content because of their spatial characteristic.

3 Methodology

Overview of the Approach. In this project, we propose to develop a flexible platform that enables both passive and active presentation modalities as well as the ability to customized presentation through a separate base of rendering rules. The rendering rules will be used to determine how the presentation MathML content will be rendered.

The mathematical expression will be aurally rendered from left to right term by term (*passive navigation*). The user can switch into another navigation scheme, *active navigation,* using the keyboard. In the active navigation the mathematical expression is navigated as an expression tree. The user can move backward and forward within the expression tree using keyboard shortcuts. S/he can switch between the two navigation schemes at any point in time, and the switch maintains the context of the navigation. Active and passive navigation will be combined (Fig.1) to give the user more flexibility when navigating mathematical content.

The goal of this work is to develop a system that provides the user with interactive navigation with an easy to update rendering rules feature. The math content in the web page will be rendered according to the rendering rules. The math content will be processed by the *rendering engine* which will apply the rendering rules from the *rule engine*.

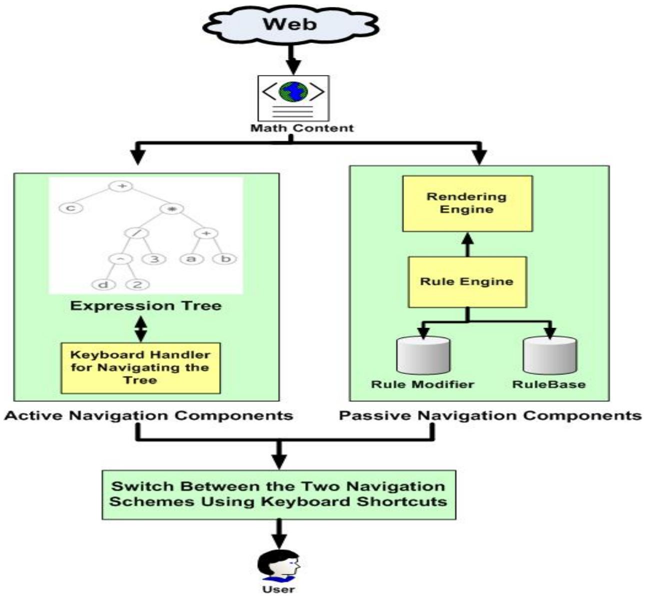
Web
Math Content
Expression Tree
Keyboard Handler for Navigating the Tree
Rendering Engine
Rule Engine
Rule Modifier
RuleBase
Active Navigation Components
Passive Navigation Components
Switch Between the Two Navigation Schemes Using Keyboard Shortcuts
User

Fig. 1. Overall Framework

Passive Presentation. Passive presentation is a navigation scheme in which sequential reading of the mathematical expression is applied (i.e. the expression is read from left to right). The passive modality is effective when dealing with expressions that have a low level of complexity (e.g., not very long and with not many levels of nesting). In order to make the passive presentation effective, lexical indicators and audio cues are added to the aurally rendered content. Offering content with speech prosody can help the user in grasping some characteristics of the mathematical content like nesting, grouping, and changing of terms inside the expression.

These features are added to the spoken text by including them in the rendering rules. The rendering rules contain several features for the spoken text. This includes pauses, speech speed, start and end messages for structural content. Rendering rules also can insert visual features to the rendered mathematical content – enabling the management of synchronized visual and aural presentation.

Each MathML tag has its own set of rules for rendering the mathematical expression. Each MathML element can admit more than one rendering rule, allowing us to modify the rendering depending on the context in which the element appears. The visual rendering allows the user to change the font colour, font style, font size and background colour of the MathML element. The audio component provides the user with the ability to change audio messages, audio speed, and audio pauses.

A rendering rule has three main components: *condition part*, *effect part*, and *MathML matching part*.

```
[MathML Element]
Rule 1: if <condition part> then
                        audio: <audio rendering effect>
                        visual: <visual rendering effect>
Rule 2: …
….
```

The MathML matching part identifies what MathML element is described by the rule (i.e., MathML tag name or text inside the MathML tag). Occurrences of the MathML element in a formula will trigger the application of the rendering rule. The condition part can be used to apply specific effects in pre-defined conditions, such as considering the size of the subexpressions. For example, the rendering rule for the **mfrac** element has two presentation effects, depending on whether the numerator and denumerator contain nested subexpressions. This allows to distinguish between cases like ½, rendered as *"one over two"*, and (x+1)/(2*Y), rendered as *"fraction X plus one divided by two times Y end fraction"*. Specifying how MathML element will be aurally or visually rendered can be done using the effect section (e.g. how many pauses to add after specific element, speed of the aurally rendered element, and font color of the element). Observe that all the aspects of prosody and insertion of lexical indicators are performed by the rendering rules.

Active Presentation. According to the WCAG 2.0 [17], providing multiple views for the content helps the user in understanding the content. Also, the guidelines mention the importance of providing the user with more than one way for navigating the content.

Active presentation is a navigation scheme in which the expression tree of the mathematical formula is interactively navigated (i.e., the user can choose in which way s/he wishes to continue navigation). This modality gives the user more focus on the mathematical content that s/he is trying to understand. The user can operate on sub-expressions, incrementally understand their structures, and eventually skip complete sub-expressions during navigation. Active presentation also facilitates the "backtracking" process – by making this dependent on the structure of the expression.

Integrating Active and Passive Presentation Modalities. The proposed framework allows the combination and interaction between passive and active presentation. Using this system the user can switch between the two modalities without losing the context of the content – i.e., when the user switches from passive to active navigation or vice versa s/he will continue from the current term in the mathematical expression and move to the next term. This is convenient to allow the user to passively listen to smaller sub-expression, and limit the active navigation to the upper layers of the expression.

4 System Design

The system is composed of four components: the MathML parser, the rendering rules parser, the rendering rules base, and the keyboard events handler. The components are integrated with FireVox. In our work we used the TTS JavaScript library as a speech synthesizer. The user requests a web page using the Firefox web browser; the browser retrieves the web page and the source code of the web page will be available to our components in the form of a DOM tree. When the web page is loaded the rendering rules will be retrieved from the XML file and the rendering rules parser will parse the file in order to identify rules that are relevant to the documents in hand. The web page source code DOM tree is parsed using the JavaScript DOM interface, and when we encounter the math tag the presentation MathML parser will use the rendering rules to represent the mathematical content both aurally and visually. The extension will automatically continue the reading of the mathematical expression using the passive presentation. The user can switch between modalities using a keyboard shortcut which will be sent to the keyboard events handler. If the user switches into the active navigation modality s/he can browse the mathematical expression tree (left, right, parent, and root) using a keyboard shortcut.

Rendering Rules. The rendering rules are stored in XML format. These rules will be matched with the mathematical expression, and they will provide information on how to render the expression. By keeping the rule base separate from the rule parser, it becomes possible to have user-dependent rules and integrate alternative presentation strategies.

The rendering rules have four main parts: presentation MathML attributes, rendering conditions, audio rendering effects, and visual rendering effects. The rendering rules are designed in such a way that each tag in the presentation MathML has its own set of rules. For example, the simplest rule for the **mfrac** element is:

```
<rule>
          <mathMLTag>mfrac</mathMLTag>
          <tokenText></tokenText>
          <numberOfArguments>2</numberOfArguments>
          <condition> <lengthOfExpression></lengthOfExpression>
                      <depthOfNesting></depthOfNesting>
                      <numberOfOperations>
                              <rendering1>1</rendering1>
                              <rendering2>5</rendering2>
                      </numberOfOperations>
          </condition>
          <audio>
                    <audioMessage>
                              <rendering1>divided by</rendering1>
                              <startMsg>fraction</startMsg>
                              <endMsg>end fraction</endMsg>
                    </audioMessage>
                    <audioSpeed>
                              <rendering1>0</rendering1>
                              <rendering2>1</rendering2>
                    </audioSpeed>
                    <audioPauses>
                              <rendering1>2</rendering1>
                              <rendering2>1</rendering2>
                    </audioPauses>
          </audio>
          <visual>
                    <fontColor>green</fontColor>
                    <fontStyle>normal</fontStyle>
                    <fontSize>12</fontSize>
                    <backgroundColor>white</backgroundColor>
          </visual>
</rule>
```

The presentation MathML tag name is presented inside *mathMLTag* tag in the rules. The text inside this tag will be matched with the MathML tag name inside a web page in order to use this set of rules to render that tag. The *tokenText* can be used in the case of a token tag (e.g., MO or MI tag) – and the *tokenText* will be used to specifically identify a set of rendering rules for the mathematical symbol or operation. The *numberOfArguments* tag in the rules is used to classify presentation MathML tags according to the number of arguments used within the element. This will help in identifying whether we will use the *startMsg* and *endMsg* elements; for example **mfrac** is a 2-argument tag which uses a start message and end message at the beginning and end of the fraction.

The conditions applied when rendering the presentation MathML is presented inside the *condition* tag in the rules. Each one of the tags can have a set of rendering conditions. The rendering conditions can be used to customize presentation depending on the complexity of the mathematical expression. The set of conditions used are inside the tags (depth of nesting of the element, size of following sub-expressions, and number of operations inside the element). Inside each one of the conditions the value represents the limit of selecting between different renderings.

In the above example the condition for the number of operations has two values (1 and 5) which will specify how this MathML tag will be rendered. If the number of operations inside this tag is greater than 1 and less than 5 then apply what is in *rendering1*. In the case of *audioMessage* it will be applied for all the number of operations values as we have only one rendering for it. The *audioSpeed* used will be 0 and 2 *audioPauses* will be used. If the number of operations inside this tag is greater than or equal 5 then apply what is in *rendering2*.

The presentation MathML element will be rendered according to the audio and visual sections of the rendering rule. The audio message can have several attributes, such as audio message, start audio message, end audio message, audio speed, and audio pauses. The visual attributes (font colour, font style, font size and background colour) and audio attributes are applied according to the rendering conditions.

Rendering Rules Parser. The rendering rules parser will parse the XML rule base when the web page is loaded. The parser is developed in JavaScript and the XML file is parsed as an XML DOM object. The rules that are identified as relevant are saved in a structure, indexed by MathML elements; the structure will be accessed during the rendering process.

Presentation MathML Parser. The web page is parsed using a DOM parser. When the math element is encountered the whole DOM tree will be sent to the presentation MathML parser. This parser will match each MathML tag in the rendering rules with what we have in the current web page. When we reach to the specific MathML element our system will start collecting information about the mathematical expression (e.g., size of the subexpressions). The collected information will be matched with the rendering conditions in order to present the mathematical expression both visually and aurally.

These different rendering conditions will select which audio and visual presentation to use. Using this information can help in determining the complexity (e.g., depth of nesting of the element, size of following sub-expressions) of the mathematical expression and the rules will be applied according to the complexity of the mathematical expression on the web page and the context use of the elements.

For example, for several operators a condition on the number of operations in the subexpressions is introduced to decide whether parentheses should be spoken. If we have the mathematical expression **tan(x)** the audio rendering will be *"tangent of x"*, but if we have the mathematical expression **tan(x/2 + y)** the audio rendering will be *"tangent of open x over two plus y close"*.

Keyboard Events Handler. The user will use keyboard shortcuts to switch between the two navigation schemes we have in the system, and to control the active navigation process. These keyboard shortcuts are managed by a keyboard event handler which will aurally change the way the mathematical content is presented.

5 System Evaluation

To evaluate our system, FireVox with multi modality, we navigated a set of web pages (e.g., an online calculus book) with significant MathML content.

Our system can selectively add specific audio cues (e.g. open, close for parenthesis. Fraction, end fraction for mfrac element) by applying rendering rules conditions.

The first example is the following presentation MathML: *cos(*θ^2*/2)-tan(*θ*).* This presentation MathML will be audio rendered using FireVox as

"C O S L per begin fraction theta sup 2 base over two r per minus T A N L per theta R per."

But in our system it will be audio rendered as

"Cosine of open fraction theta power two over two end fraction close minus tangent of theta."

Notice that, in the second sub-expression (i.e., tan(θ)), our system did not read open and close parenthesis, since the complexity of the subexpression is below the set threshold (it should include an additional operation). On the other hand, the first sub-expression (i.e., cos(θ^2/2)) our system explicitly speaks the parentheses.

From the above example we can see that the new system can correctly read the common abbreviated mathematical functions (i.e., tan and cos as tangent and cosine). Our system can read other common mathematical functions (e.g. sin, cos, tan, log, ln ...), having matching rules for these different cases.

The second example that shows the difference between the audio rendering of our system and FireVox are the following:

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}$$

By using FireVox this presentation MathML encoding will be audio rendered as

"A sub zero base plus begin begin begin begin fraction one over over over over, A sub one base plus begin begin begin fraction one over over over, A sub two base plus begin begin fraction one over over, A sub three base plus begin fraction one over A sub four base, End fraction, End end fraction, End end end fraction, End end end end fraction."

But using our system the audio rendering will be

"A sub zero plus fraction one over, a one plus fraction one over , a two plus fraction one over, a three plus one over a four, End fraction, End fraction, End fraction."

It is not easy for a user to comprehend the meaning of the previous expression using the audio the result from FireVox, as it overloads the audio with contextual reinforcements. In our approach, we have designed rules to reduce some of such overhead, based again on the size of the subexpressions.

In the following example we will show how the user can interactively audio navigate the mathematical expression using our system.

$$100x + \frac{\sqrt{3-x}}{4} - \frac{\pi}{9} = x^3$$

When the user navigates the above formula using interactive navigation s/he will hear the root element of the expression tree, which is "=", and then the user can select to move to the left or right sub-expression. The user can go back and forth between different sub-expressions until s/he comprehends them. If the user switches back to the passive presentation the reading will continue from the current location on the expression tree and move linearly from left to right in the mathematical expression.

Our system introduces presentation MathML elements that are not recognized in FireVox (e.g., matrix and table elements).

6 Conclusion and Future Work

Studying mathematical contents is not an easy task especially for the visually impaired. In this work we presented a flexible approach when rendering web mathematical expressions. The methodology relies on the integration between active and passive presentation schemes. The use of lexical and audio indicators is combined with two navigation strategies. Providing the users with more than one navigation modality will help them in comprehending the mathematical content much better.

The results show the importance of active reading of mathematical content. Also it points to the importance of analyzing the complexity of mathematical expressions. The new system renders the mathematical expressions in minimum spoken text while the content is unambiguously rendered to the user. In the future we will expand the rendering rules to handle haptic. We can offer users different templates for rendering rules that can be applied for different kinds of disabilities. Introducing the concept of rendering rules will not only help making the web more accessible for the visually impaired users but it will also improve the universal accessibility of the web by applying rules for other kinds of disabilities.

Acknowledgments. The research has been partially supported by NSF grants 0220590 and 0754525.

References

1. Lighthouse Int. Prevalence of Vision Impairment (2009), `http://www.lighthouse.org/research/statistics-on-vision-impairment/prevalence/`
2. Pontelli, E., Karshmer, A.I., Gupta, G.: Mathematics and Accessibility: a Survey. In: The Universal Access Handbook. Taylor & Francis, Abington (to appear)
3. Stevens, R.D., et al.: Access to mathematics for visually disabled students through multimodal interaction. In: Human-Computer Interaction (1997)
4. Fitzpatrick, D.: Speaking Technical Documents: Using Prosody to Convey Textual and Mathematical Material. In: Miesenberger, K., Klaus, J., Zagler, W.L. (eds.) ICCHP 2002. LNCS, vol. 2398, p. 494. Springer, Heidelberg (2002)
5. Fitzpatrick, D.: Mathematics: How and What to Speak. In: International Conference on Computers Helping People with Special Needs (2006)
6. Ferreira, H., Freitas, D.: Enhancing the Accessibility of Mathematics for Blind People: the AudioMath Project. In: Miesenberger, K., Klaus, J., Zagler, W.L., Burger, D. (eds.) ICCHP 2004. LNCS, vol. 3118, pp. 678–685. Springer, Heidelberg (2004)
7. Ferreira, H., Freitas, D.: AudioMath: Towards Automatic Readings of Mathematical Expressions. In: Human Computer Interaction International (2005)

8. Gaura, P.: REMathEx - reader and editor of the mathematical expressions for blind students. In: Miesenberger, K., Klaus, J., Zagler, W.L. (eds.) ICCHP 2002. LNCS, vol. 2398, p. 486. Springer, Heidelberg (2002)
9. Karshmer, A.I., et al.: Helping visually impaired students in the study of mathematics. In: IEEE Frontiers in Education Conference (1999)
10. Awde, A., et al.: Complexity of Mathematical Expressions in Adaptive Multimodal Multimedia System. Int. Journal Computer & Info. Science and Eng. (2008)
11. FireVox: A Screen Reading Extension for FireFox (2009), http://firevox.clcworld.net/
12. Raman, T.V.: Audio Systems for Technical Reading, in Computer Science. Cornell University, New York (1994)
13. Reddy, H., et al.: Listener-Controlled Dynamic Navigation of VoiceXML Documents. In: Miesenberger, K., Klaus, J., Zagler, W.L., Burger, D. (eds.) ICCHP 2004. LNCS, vol. 3118, pp. 347–354. Springer, Heidelberg (2004)
14. Soiffer, N.: Advances in Accessible Web-based Mathematics. In: CSUN International Conference on Technology and Persons with Disabilities (2005)
15. Gillan, D.J., et al.: Cognitive analysis of equation readings: application to the development of the MathGenie. In: Miesenberger, K., Klaus, J., Zagler, W.L., Burger, D. (eds.) ICCHP 2004. LNCS, vol. 3118, pp. 630–637. Springer, Heidelberg (2004)
16. Barraza, P., et al.: A cognitive analysis of equation reading applied to the development of assistive technology for visually-impaired students. In: Human Factors and Ergonomics Society Annual Meeting (2004)
17. Web Content Accessibility Guidelines (WCAG) 2.0 (2009), `http://www.w3.org/TR/WCAG20`