Content-based Navigation within Mathematical Formulae on the Web for Blind Users and its Impact on Expected User Effort

Luiz Felipe da Paixão Silva, Antonio Armando de Oliveira Barbosa, Evelise Roman Corbalan Gois Freire, Paula Christina Figueira Cardoso, Rafael Serapilha Durelli, André Pimenta Freire
Federal University of Lavras
Lavras, MG, Brazil
luizfelipe_p@hotmail.com,barbosaneto23@hotmail.com,evelise.freire@dex.ufla.br,\protect\T1\textbraceleftpaula.cardoso,rafael.durelli,apfreire\protect\T1\textbraceright@dcc.ufla.br

ABSTRACT

Learning Mathematics presents many challenges for blind students. The support for reading and navigating within web-based mathematical content is still limited in screen reader software, specially for languages such as Brazilian Portuguese. Lengthy and complex formulae, in particular, may demand significant effort from users who navigate only by keyboard and with linear feedback sound. Current screen readers provide limited support with semantic navigation based on content type. In this paper, we present a study involving the implementation of an initial prototype screen reader as proof-of-concept based on ChromeVox, aggregating new features for direct navigation with semantic mathematical elements, such as fractions, radicals and summation indexes. We also performed a study on the expected user effort of this approach, comparing it with the resources provided by other existing screen readers, namely JAWS, NVDA (Non-Visual Desktop Access). The estimation was performed using task models based on GOMS (Goals, Operators, Methods and Selection Rules) and KLM (Key-Level Model). The results showed that it is feasible to implement forms of navigation based on problem solving, because according to task models there was an expressive reduction in the estimated time to complete tasks.

CCS CONCEPTS

• **Human-centered computing → Accessibility design and evaluation methods**;

KEYWORDS

Screen reader navigation, Mathematics, Task models

ACM Reference Format:
Luiz Felipe da Paixão Silva, Antonio Armando de Oliveira Barbosa, Evelise Roman Corbalan Gois Freire, Paula Christina Figueira Cardoso, Rafael Serapilha Durelli, André Pimenta Freire. 2018. Content-based Navigation within Mathematical Formulae on the Web for Blind Users and its Impact on Expected User Effort. In *Proceedings of 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI'18).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Assistive technologies are an essential resource people with visual disabilities to have appropriate access to education,e specially in fields such as Mathematics and Sciences. Technologies to aid learning of such contents should provide appropriate means to help perform complex tasks, which are inherent of such areas.

Certain difficulties can be be even more pronounced in the case of mathematical problems, particularly those involving algebraic notation. The complexity of mathematical problems can be more difficult for blind users than for sighted users due to the use of visual symbols in algebraic structures, which is per se very challenging to students in general. The lack of appropriate technologies to help read and learn mathematical content can pose serious challenges to blind students in Mathematics classes.

Whilst sighted students solve mathematical problems by means of written language, and by blind students using of Braille or spoken text. However, reading mathematical content in Braille can limit the experience of blind users with mathematical content. This is mainly due to fewer possibilities to "navigate" between lines, besides the need to memorize terms and partial results [6]. For this reason, the use of screen reader software with speech synthesis, commonly used for using general-purpose computer software, can have more possibilities to blind users.

Although previous studies have presented specialized stand-alone mathematical reading software for blind users, such as Lambda [5] and rules for mathematical reading in English such as MathSpeak® [8], the availability of embedded features to read mathematical content is very recent in screen readers, particularly those distributed in Brazil, such commercial software JAWS[1], and free software systems NVDA[2] and ChromeVox[3]. Although those screen readers are able to read general-purpose text in Brazilian Portuguese, to date, none of them provides support for reading Maths content in Portuguese.

Considering that screen reader software synthesizes speech in a sequence, it is very important that blind users are able to use

[1]JAWS - Available at http://www.freedomscientific.com/, Maths support since version 16, in 2014
[2]NVDA - Non-visual Desktop Access - Available at http://www.nvaccess.com, Maths support since 2015, integrating the external plug-in MathPlayer [11, 12]
[3]ChromeVox - Available at http://www.ChromeVox.com, Maths support since 2014 [13]

features to speed up their navigation to reach the content. On web sites, for example, users frequently use navigation by reading the headings of a page first, and then reading into detail the content they expect to be most relevant. According to the latest survey performed by the organization Web Accessibility in Mind (WebAIM) in 2017 [16], more than 67% of the 1792 screen-reader users surveyed in this study use this technique when navigating in long pages. Screen readers also support navigation to reach other elements based on their semantics, such as lists, paragraphs, images, forms, and others.

However, none of the screen readers with support to reading Mathematical content currently provides similar features to navigate within complex Mathematical formulae based on semantic mark-up of their elements, in a similar fashion to what occurs on web pages. Thus, there is little known about how such strategies could help improve the expected user effort to read such formulae if such resources were available.

In a previous study performed by some of the authors [7] of this paper, we performed a comparison of screen readers with mathematical reading available in terms of the expected effort from users on tasks with the navigation strategies offered by them.

In the present paper, we report on the implementation of a proof-of-concept prototype with traditional formulae navigation features available on existing screen readers, and a selection of new features to allow navigation based on mathematical semantical elements and special internal elements, such as indices, parts of fractions and other structures, using MathML (Mathematical Markup Language). The implementations were based on the open-source software Chromevox. Following this implementation, we analyzed task models using functions as well as additional functions to the open source software ChromeVox and to compare with the performance of the others using task models GOMS (Goals, Operators, Methods and Selection Rules) and KLM (Key-Level Model) to compare the expected user effort in best conditions, using the implemented features, and navigation using the existing approaches.

This remainder of this paper is organized as follows. Section 2 presents the theoretical background, with the basic concepts and related work. Section 3 details the methods, with information regarding the tools and techniques used. In Section 4, we present the results obtained and discussion. Finally, Section 5 presents conclusions and future work.

2 THEORETICAL BACKGROUND

2.1 Mathematical Markup Language

MathML (Mathematical Markup Language) [3] is a language based on XML (eXtensible Markup Language) for the markup of mathematical content. MathML is also composed of tags, and represents data in a well-structured manner. MathML has two main strands of notations: MathML presentation and MathML content.

MathML presentation is used to describe the structure of mathematical expressions, with focus on the way it is rendered on the user's screen. To represent mathematical expressions visually we use some elements such as: (1) <mi> - identifier, (2) <mn> - number, (3) <mo> - operator, fence, or separator, (4) <mtext> - text, (5) <mspace> - space, (6) <ms> - string literal.

Other presentation elements are called layout schemata. These elements only contain other elements as content and are not token elements. We listed a selection of examples of such layout elements as follows: (1) <mrow> - groups any number of sub-expressions horizontally, (2) <mfrac> - composes a fraction from two sub-expressions, (3) <msqrt> - composes a square root (radical without an index), (4) <mroot> - composes a radical with specified index, (5) <mfenced> - surrounds content with a pair of fences, (6) <menclose> - encloses content with a stretching symbol such as a long division sign, (7) <msub> - attaches a subscript to a base, (8) <msup> - attaches a superscript to a base, (9) <msubsup> - attaches a subscript-superscript pair to a base, (10) <munder> - attaches an underscript to a base, (11) <mover> - attaches an overscript to a base, (12) <munderover> - attaches an underscript-overscript pair to a base.

The other strand of MathML specification is called content MathML, which focuses on the semantics or meaning instead of the expression layout, i.e., tags are used that denote operations and how they should be applied. Content MathML contains a tag called "apply", where the first tag inside it is the function that will be applied and the other tags are the parameters and operators.

In addition to these tags, there are many others available in the specification [3]. Having access to these tags, it is possible to devise navigation techniques to make it possible for blind users reach specific content, such as the numerator of a fraction, as sighted users do when solving exercises.

2.2 Task modelling and KLM

Task modeling is defined as a step-by-step description of tasks that must be fulfilled by a user in order to achieve some objective. Task models can be used to create interfaces, and also to to analyze and evaluate the interactivity of applications [1].

Over the years, several approaches to task models have been developed. One of the most widely used methods in the literature is GOMS, defined by its authors as "a set of Goals, a set of Operators, a set of Methods for achieving the goals, and a set of Selections rules for choosing among competing methods for goals" [2], and how to use operators, methods and selection rules.

For this study, we used the model Keystroke-Level Model or simply KLM [2] in conjunction with GOMS. KLM uses only the pressed keys, mouse movements, and mouse keys as a way of evaluating and analyzing a task in terms of observable actions, as well as mental operators and waiting times. The goal of using KLM is to have an estimate of the expected time and effort that users would have in a task.

2.3 Related Work

In an earlier study, Stoeger *et al.* [14] investigated requirements to make navigation and reading of mathematical content possible to blind students. Among the main results, that paper listed issues with the inaccessibility of digital content and how screen readers and other technologies should be provide further features to overcome those difficulties.

Regarding the use of task models, we encountered few reports in the literature concerning the use of assistive technologies by blind users on the Web. The main studies encountered in this respect are described as follows.

Schrepp and Fischer [10] performed a study on the accessibility of pages with higher number of interactive elements. To this end, they used a GOMS model derive estimations of the time that a user would take to perform a task in applications in which keyboard and mouse are used with high frequency. The authors concluded that by using average values estimated by GOMS, it was possible to have an initial estimation to evaluate different interfaces.

In another older study, Edwards [4] verified if it was possible to adapt the visual interfaces of an application built with WIMP (Windows, icons, menus and pointers) to an auditory form that could be used by blind people. For that, he adapted a piece of WIMP software to create a word processor called soundtrack. With an analysis of the number of keystrokes and time-on-task, that paper concluded that the development of assistive technologies for blind people at the time had many challenges related to the complexity of interaction.

In another study, Trewin *et al.* [15] investigated the need for a reliable prediction of effort on tasks by means of keyboard-level modelling for an efficient analysis of the accessibility of screen readers for disabled users. They developed a tool developed that enabled modelling of screen reading interaction approaches, as well as exploring action sequences and the interaction of less experienced users, not so common in task model approaches.

According to Schrepp [9], an efficient keyboard access to Web sites is highly important for many groups of disabled users. In their study, the authors evaluated a number of web sites, in order to show the main challenges encountered by users with visual disabilities, who use keyboard-only navigation to access the content of Web sites. For this, GOMS modelling was used to obtain quantitative values of keyboard navigation performance. The paper pointed out the main design errors that jeopardize navigation on keyboard.

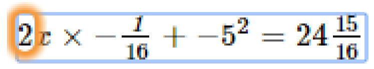
3 METHOD

3.1 Study design

The present paper involved the implementation of a proof-of-concept prototype based on the open-source software ChromeVox. The new version was called ChromeVox - NavMatBR, following the acronym of the main overarching research project in which it was developed. The prototype included new keyboard commands to access specific elements of three mathematical structures: fractions, summations and radicals.

Following the implementation, in order to compare the effort estimation for blind users when browsing web-based formulae, we selected three expressions using the three selected structures in the proof-of-concept, in which users would need to identify parts of specific expressions.

Following this, task models were created considered a best-case scenario in which blind users would have full command of their screen readers and would have well-developed abilities to identify parts of expressions. This way, the models portrayed an estimation of the minimum effort that would be required using the resources available in the screens readers analyzed. Although we recognize that this does not represent what would be done by all blind users, this model with the best-case scenario represents the complexity



$$2x \times -\frac{1}{16} + -5^2 = 24\frac{15}{16}$$

Figure 1: Character navigation using ChromeVox

of the tasks in terms of the way the resources available were implemented.

The models were created considering the commands for keyboard navigation within mathematical formulae available in JAWS, ChromeVox, NVDA, and the new prototypf ChromeVox - NavMatBR.

From the task models, we were able to perform analyses of the expected effort in the different screen readers and navigation approaches.

3.2 Existing screen readers analyzed

For this study, three screen readers were used to compare with the implementation of the proof-of-concept prototype of ChromeVox - NavMatBR, described as follows.

ChromeVox version 53.0.2784.5: this screen reader is an open-source plugin freely available via the Chrome browser, and can be used in multiple platforms. Being open source, it allowed for the modifications carried out in the proof-of-concept prototype described in this paper.

ChromeVox allows users to edit their command keys for any type of action, and by default some commands are defined with the addition of other keys. For the evaluation of the screen reader only some basic commands were used, considering the default settings:

- TAB key - goes to the next focusable item.
- ChromeVox command - standard command key to start commands, being the combination of the Shift + Alt keys by default.
- ChromeVox command + N > M - Go to the next mathematical expression found on the page.
- ChromeVox command + equal - increases the granularity, i.e. causes the focus on the expression to be changed by levels such as the characters, for example.
- ChromeVox key + down arrow - navigates forward or to the next element.

The main features provided by the original ChromeVox version aimed to allow reaching mathematical elements and reading characters within the mathematical formula. Figure 1 shows an example of reading in a higher level of granularity. Although the software the option of changing granularity, this feature was not used in the task models due to the difficulty of identifying operators that are not read by the reader. One limitation of the screen reader is that it only works within the Chrome browser environment

The Non-visual Desktop Access (NVDA) 2016.4 is a free screen reader, compatible with the Windows operating system. NVDA also has a series of commands, and allows for powerful resources to navigate in Web elements, such as navigating across headings, paragraphs, links, and others. However, for the particular case of this analysis, only basic commands to access mathematical elements are available. Mathematical reading in NVDA is done by access to an external plug-in, MathPlayer, and NVDA has no control over the navigation within formulae, which is done by the plug-in. In

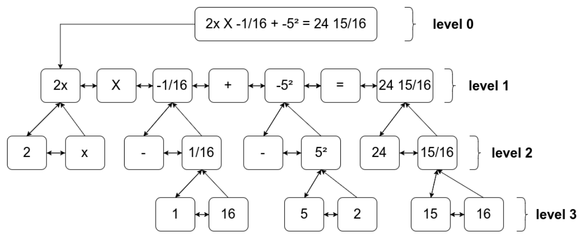


Figure 2: Tree and the levels where the JAWS reads the partial expressions

order to re-read a part of a formula, it is necessary to return to its beginning and read it all over again. Following are some basic commands from NVDA used in the task analysis in this research:

- NVDA key - Insert of the numerical keypad, Extended Insert or the Caps Lock key can be used as your modifier keys.
- NVDA + tab - focuses on the current item.
- Down arrow - moves to next element on page.
- Up arrow - returns to previous element on page.

Although NVDA is a powerful and widely used software, in the case of mathematical content it does not provide support or means for internal navigation.

The Job Access With Speech (JAWS) version 17.0 is currently the most popular screen reader worldwide [16], is a commercial software, and is compatible with the Windows operating system. The screen reader presents relevant features compared to its competitors regarding reading of mathematical content on Websites. JAWS also has a number of commands to explore different parts of Web pages. However, in this paper, we focused on the commands used to reach a formula within a page and on the commands used to navigate within formulae, shown as follows:

- Down arrow - focuses to the next page element.
- Enter key - open the mathematical formula viewer when finding an expression.
- Right arrow - read the next character of the word where the system focus is, also used to navigate within the mathematical formulas viewer.

Among all readers, JAWS provides the most advanced support to mathematical content. The software allows users to navigate within mathematical formulae, so that it becomes a mathematical expression in a tree, as shown in Figure 2. we can also see in the image that the arrows indicate the reading stream of expressions.

3.3 ChromeVox-NavMatBR

The implementation of the initial proof-of-concept prototype of ChromeVox-NavMatBR was based on the available screen readers cited in the previous section, especially the more advanced features of tree navigation in JAWS.

Further to the previously existing features, the prototype implemented a selection of navigation techniques to traverse parts of algebraic expressions.

Thus, new features were added to implementation of ChromeVox to assess the performance of the modified software. By doing this analysis, we hoped to help guide new future of navigation in formulae. One of the functionalities implemented was a search for fractions within an expression, identification of lower and upper limits in summation and navigating withing the parts of a radical, as well as the internal navigation in algebraic expressions similar to JAWS's tree view (but without the visualization).

3.4 ChromeVox-NavMatBR implementation

ChromeVox-NavMatBR was implemented based on the open-source implementation of ChromeVox, described in Section 3.2.

In total five new features have been implemented. Most of the commands used are subject to change. For navigation within parts of the formulae, only the activation and deactivation keys can be modified, the other keys in this case, the directional arrows on the keyboard will not be able to to be modified, since for a better performance the smaller number of keys pressed to perform a function is ideal.

The internal parts navigation commands are defined as:

- Enable split enter + shift navigation.
- Exit the internal navigation escape key.
- Navigate right inside the expression right key.
- Navigate left inside the expression left key.
- Navigate to a level above the up key expression.
- Navigate to a level below the expression key down.

To find fractions within the mathematical expressions, the following fraction command was created with the following keys:

- Cvox + N>D to go to the next fraction.
- Cvox + P>D to go to the previous fraction.

Similarly to the feature that finds fractions in a mathematical expression, an extra functionality was implemented to find the limits of an existing summation in the expression. We also implemented a feature to find the next and previous summation on the page, using the commands defined as Cvox + N> S to go to the next found sum and Cvox + P> S to go to the previous summation.

Another functionality implemented was the numerator and denominator identification. This functionality was implemented as follows: when the numerator or denominator identification command is pressed and the navigation focus is in a Mathml tag called mfrac, we update the focus to the first child of mfrac that will be the numerator or denominator, the definition of which tag will be focused will depend exclusively on the user's desire. The commands defined to identify the numerator and denominator are respectively shitf + N and shitf + D.

A last command was implemented that has the objective of finding the roots in an equation, the command defined for this function was Cvox + N> R to search the next root and Cvox + P> R to return to a previously found root.

In Table 1, we can see all commands implemented.

3.5 Task modelling

In order to analyze and evaluate the expected effort demanded from people with visual disabilities to identify parts or even solve mathematical expressions, we used the two predictive models GOMS and KLM.GOMS was used to model the goals and actions of the users that should be taken to perform simple tasks in the expressions chosen for testing. The proposed tasks were:

(1) Verify if the expression has any numerator whose value is fifteen. This task was applied to the expression

Table 1: All commands implemented in ChromeVox-NavMatBR

Commands	Functionalities
Shift + enter	Enable internal navigation
Esc	Disable internal navigation
Rigth arrow	Navigate to the right side of the expression
Left arrow	Navigate to the left side of the expression
Up arrow	Navigate one level above the expression
Down arrow	Navigate one level below the expression
Cvox + N >D	Go to the next fraction
Cvox + P >D	Go to the previous fraction
Cvox + N >S	Go to the next summation
Cvox + P >S	Go to the previous summation
Shift + N	Go to the numerator of the expression
Shift + D	Go to the denominator of the expression
Cvox + N >R	Go to the next root of the expression
Cvox + P >R	Go to the previous root of the expression

$$2x \times -1/16 + -5^2 = \ 24\ 15/16 \tag{1}$$

(2) Verify the upper limit in the summation

$$\sum_{m=1}^{3} \sum_{n=1}^{m} \sin(x^m + y^n) \tag{2}$$

(3) Verify the index of the second root in the expression

$$A = \sqrt{12^2} + \sqrt[3]{125} \tag{3}$$

For the first task, users would have to find the mathematical formula in a Web page, and after that they should find the fraction in the formula and verify if the numerator was fifteen.

In the second task, users would have to find the formula on the page and find the second summation. Then, they would have to check the value of the upper bound.

For the last task, the user would have to find the formula he wants on the web page and check its second root's index.

We considered that after each keyboard operation, blind users would have to perform a mental operation to assess whether the content they had listened to was relevant to their task.

For the planning of the tasks, we defined the use of three expression being 1 and 2 in written MathML (Mathematical Markup Language) [3]. As an example, the MathML code for Expression 1 is presented in Listing 1.

It is important to highlight that, in order to use predictive models such as GOMS, the models will be limited to conditions in which users already know what the tasks are, what they have to do and also that they would have previous knowledge about Mathematics, in this particular case.

Finally, the KLM model was used to estimate the time required by users to complete the selected tasks. In this work, only the operators K (keystroke), M (mental operation) and W (waiting for system response) were used. Considering that the models were targeted at blind users, only keyboard operations would be used as input methods to perform the defined tasks.

3.6 Data analysis

The task models and analyses presented in this paper were performed following the same method performed in the comparison carried out in a previous study from the authors' research group [7], now including new tasks and the new implemented proof-of-concept prototype with new navigation strategies. The tasks to be analyzed encompassed the steps to navigate a Web page, find an expression and obtain information from the expression by means of the terms analyzed. Given the tasks and their actions, we used KLM to evaluate each task according to the number of operations that needed to be carried out.

Listing 1: MathML code for expression 1

```
<math xmlns="http://www.w3.org/1998/Math/MathML" >
      <mn>2</mn>  <mo>&#x2062;</mo>     <mi>x</mi>
      <mo>&#xD7;</mo>    <mo>-</mo>
      <mfrac>
            <mrow>  <mi>1</mi></mrow>
            <mn>16</mn>
      </mfrac>
      <mo>+</mo>
      <msup>
            <mrow> <mo>&#x2212;</mo>    <mn>5</mn>  </mrow>
            <mn>2</mn>
      </msup>
      <mo>=</mo>    <mn>24</mn>      <mo>&#x2064;</mo>
      <mfrac>
            <mn>15</mn>
            <mn>16</mn>
      </mfrac>
</math>
```

Unfortunately, we could not find previous research with reference values for times taken by blind users to perform tasks using screen readers. Were are aware that the time taken by blind users can be considerably longer than for sighted users. However, in order to have some means of comparing the values numerically, we used the reference values for KLM operations defined by Card *et al.* [1].

The execution time of a task was the sum of the time of all the operators involved to complete the task. From that estimation we calculated the sum of all the operators used, being K (keystroke), M (mental operation) and W (wait for system response).

The W operator was used mainly for cases in which users would wait until the screen reader finished reading a word. We used the value of 1 second per word as an average estimation. This is particularly important in the consideration of such tasks in comparison to the interaction by sighted users, as listening to spoken content takes a considerable time for blind users.

Two models of a GOMS with time analyses using KLM were used for each screen reader. By obtaining the total time for each task using the reference values presented by Card *et al.* [1], the time on each task was compared between the screen readers used.

As previously mentioned, the values do not reflect the actual time that blind users would take on tasks, but were important to enable comparisons in terms of the proportion of time difference for the navigation tasks between the evaluated screen readers.

4 RESULTS AND DISCUSSION

4.1 Task models

The tasks were modelled using the CMN-GOMS (Card, Moran and Newell GOMS), with a structure that represents a pseudo-code of the realized tasks. This was the original version of the GOMS model proposed by Card, Moran and Newell [2]. Twelve task models were created.

In Listing 2, we show the GOMS model for Task 1 (to verify if the expression has any numerator whose value is fifteen) applied on expression 1 using the screen reader ChromeVox. The total estimated number of operations on Task 1 on ChromeVox was 16 K, 25 M and 73 W, being 16 keystrokes, 25 mental operations and waiting for reading 73 words.

Listing 2: GOMS model for Task 1 using ChromeVox

```
GOAL:VERIFY IF THE EXPRESSION HAS ANY NUMERATOR WHOSE VALUE
IS FIFTEEN
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       .PRESS SHIFT + ALT + N> M    4K + w(30) + 1M
.       .       .PRESS SHIFT + ALT + N> M    4K + w(21) + 1M
.       . GOAL: FIND FRACTION IN EXPRESSION
.       .       . PRESS SHIFT + ALT + P> M  4K + w(30) + 1M
.       .       . PRESS SHIFT + ALT + N> M  4K + w(21) + 21M
.       . GOAL: CHECK THAT THE VALUE IS FIFTEEN  1M
TASK1: 16K + W(73) + 25M = 107.48
```

Listing 3 shows the GOMS model for Task 1 for the JAWS screen reader. The total number of operations on Task 1 on JAWS was 18 keystrokes, 18 mental operations and waiting for 105 words.

Listing 3: GOMS model for Task 1 using JAWS

```
GOAL: VERIFY IF THE EXPRESSION HAS ANY NUMERATOR WHOSE VALUE IS FIFTEEN
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       .PRESS DOWN ARROW       1K + W(2) + 1M
.       .       .PRESS DOWN ARROW        1K + W(5) + 1M
.       .       .PRESS DOWN ARROW       1K + W(2) + 1M
.       .       .PRESS DOWN ARROW       1K + W(35) + 1M
.       .       .PRESS DOWN ARROW       1K + W(2) + 1M
.       .       .PRESS DOWN ARROW       1K + W(17) + 1M
.       . GOAL: FIND FRACTION IN EXPRESSION
.       .       .PRESS ENTER            1K + W(21) + 1M
.       . GOAL: CHECK THAT THE VALUE IS FIFTEEN
.       .       .PRESS RIGHT ARROW      1K + W(1) + 1M
.       .       .PRESS LEFT ARROW       1K + W(1) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(1) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(4) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(1) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(3) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(1) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(4) + 1M
.       .       .PRESS DOWN ARROW       1K + W(1) + 1M
.       .       .PRESS RIGHT ARROW      1K + W(2) + 1M
.       .       .PRESS DOWN ARROW       1K + W(2) + 1M
TASK1: 18k + W(105) + 18M = 131.64
```

Listing 4 shows the GOMS model for Task 1 for the NVDA screen reader. The total number of operations on Task 1 on NVDA was 6 keystrokes, 22 mental operations and waiting for reading 72 words, considering all the repetitions necessary to re-read the content due to the lack of in-formula navigation.

Listing 4: GOMS model for Task 1 using NVDA

```
GOAL: VERIFY IF THE EXPRESSION HAS ANY NUMERATOR WHOSE VALUE
IS FIFTEEN
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       PRESS DOWN ARROW        1K + W(34) + 1M
.       .       PRESS DOWN ARROW        1K+W(2) + 1M
.       .       PRESS DOWN ARROW        1K + W(2) + 1M
.       .       PRESS DOWN ARROW        1K + W(16) + 1M
.       . GOAL: FIND FRACTION IN EXPRESSION
.       .       PRESS UP ARROW   1K + W(2) + 1M
.       .       PRESS DOWN ARROW        1K + W(16) + 16M
.       . GOAL: CHECK THAT THE VALUE IS FIFTEEN  1M
TASK1 : 6K +W(72) + 22M = 100.08
```

In Listing 5, we show the GOMS model for task 1, using the screan reader ChromeVox-NavMatBR whose new features was added to improve performance. The total number of operations on Task 1 on ChromeVox-NavMatBR was 18 keystrokes, 5 mental operations and waiting for reading 65 words.

In Listing 6, we show the GOMS model for Task 2 (verify the upper limit in the second sum) using ChromeVox. The total number of operations on Task 2 on ChromeVox was 12 keystrokes, 21 mental operations and waiting for reading 87 words.

Listing 7 shows the GOMS model for Task 2 for the JAWS screen reader. The total number of operations on Task 2 on JAWS was 12 keystrokes, 12 mental operations and waiting for 135 words.

Listing 8 shows the GOMS model for Task 2 for the NVDA screen reader. The total number of operations on Task 2 on NVDA was 3 keystrokes, 19 mental operations and waiting for reading 73 words.

Listing 5: GOMS model for Task 1 using ChromeVox-NavMatBR

```
GOAL:VERIFY IF THE EXPRESSION HAS ANY NUMERATOR WHOSE VALUE IS FIFTEEN
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       .PRESS SHIFT + ALT + N> M    4K  + W(30) + 1M
.       .       .PRESS SHIFT + ALT + N> M    4K + W(21) + 1M
.       . GOAL: FIND FRACTION IN EXPRESSION
.       .       .PRESS SHIFT + ALT +N>D  4K + W(5) + 1M
.       .       .PRESS SHIFT + ALT +N>D  4K + W(5) + 1M
.       . GOAL: CHECK THAT THE VALUE IS FIFTEEN
.       .       .PRESS SHIFT + N      2K+ W(4) + 1M
TASK1 : 18K+ W(65) + 5M = 76.04
```

Listing 6: GOMS model for Task 2 using ChromeVox

```
GOAL: VERIFY THE VALUE OF THE HIGHEST LIMIT OF THE SECOND
SUMMATION IN THE EXPRESSION
.       . GOAL:FIND THE DESIRED EXPRESSION
.       .       . PRESS SHIFT + ALT + N> M  4K  + W(30) + 1M
.       .GOAL: FINDING THE DESIRED SUM IN EXPRESSION
.       .       . PRESS SHIFT + ALT + P> M  4K + W(24) + 1M
.       .       . PRESS SHIFT + ALT + N> M      4K  + W(33) + 18M
.       . GOAL: CHECK THE LIMIT VALUE 1M
TASK2: 12K + W(87) + 21M = 111.36
```

Listing 7: GOMS model for Task 2 using JAWS

```
GOAL: VERIFY THE VALUE OF THE HIGHEST LIMIT OF THE SECOND
SUMMATION IN THE EXPRESSION
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       PRESS DOWN ARROW        1K + W(2) + 1M
.       .       PRESS DOWN ARROW        1K + W(5) + 1M
.       .       PRESS DOWN ARROW        1K+W(2) +1M
.       .       PRESS DOWN ARROW        1K + W(35) + 1M
.       . GOAL: FINDING THE DESIRED SUM IN EXPRESSION
.       .       PRESS ENTER     1K + W(21) + 1M
.       . GOAL: CHECK THE LIMIT VALUE
.       .       PRESS RIGHT ARROW       1K+ W(21)  +1M
.       .       PRESS LEFT ARROW        1K+ W(10) +1M
.       .       PRESS RIGHT ARROW       1K+ W(21) +1M
.       .       PRESS DOWN ARROW        1K+ W(10) +1M
.       .       PRESS DOWN ARROW        1K +1M
.       .       PRESS RIGHT ARROW       1K+ W(4) +1M
.       .       PRESS RIGHT ARROW       1K+W(3)+1M
TASK2: 12k + W(135)+ 12M = 152,76
```

Listing 8: GOMS for Task 2 using NVDA

```
GOAL: VERIFY THE VALUE OF THE HIGHEST LIMIT OF THE SECOND
SUMMATION IN EXPRESSION
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       .PRESS DOWN ARROW 1K + W(34) + 1M
.       . GOAL: FINDING THE DESIRED SUM IN EXPRESSION
.       .       .PRESS UP ARROW 1K + W(5) + 1M
.       .       .PRESS DOWN ARROW 1K + W(34)+17M
.       .GOAL: CHECK THE LIMIT VALUE  1M
TASK2: 3K + W(73) + 19M = 96.64
```

Listing 9 shows the GOMS model for Task 2 for the ChromeVox-NavMatBR screen reader. The total number of operations on Task 2 on ChromeVox-NavMatBR was 17 keystrokes, 6 mental operations and waiting for reading 72 words.

Listing 10 shows the GOMS model for Task 3 for the ChromeVox screen reader. The total number of operations on Task 3 on ChromeVox was 20 keystrokes, 17 mental operations and waiting for 102 words.

Listing 11 shows the GOMS model for Task 3 for the ChromeVox -NavMatBR screen reader. The total number of operations on Task 3

Listing 9: GOMS for Task 2 using ChromeVox-NavMatBR

```
GOAL: VERIFY THE VALUE OF THE HIGHEST LIMIT OF THE SECOND
SUMMATION IN THE EXPRESSION
.   .GOAL: FIND THE DESIRED EXPRESSION
.   .   .PRESS SHIFT + ALT + N> M   4K  + w(30) + 1M
.   .GOAL: FIND THE DESIRED SUM IN EXPRESSION
.   .   .PRESS SHIFT + ALT +N>D     4K + W(11) + 1M
.   .   .PRESS SHIFT + ALT +N>D     4K + W(11) +1M
.   .GOAL: CHECK THE LIMIT VALUE
.   .   .PRESS SHIFT + ENTER       2K + W(10)
.   .   .PRESS DOWN ARROW       1K + W(4) +1M
.   .   .PRESS RIGHT ARROW       1K + W(4)+1M
.   .   .PRESS RIGHT ARROW       1K +W(2)+1M
TASK2:   17K + W(72) + 6M = 83.96
```

Listing 10: GOMS for Task 3 using ChromeVox

```
GOAL: VERIFY THE INDEX OF THE SECOND ROOT OF the EXPRESSION
.   .GOAL: FIND THE DESIRED EXPRESSION
.   .   PRESS SHIFT + ALT + N> M        4K  + w(30) + 1M
.   .   PRESS SHIFT + ALT + N>  4K + w(21) + 1M
.   .   PRESS SHIFT + ALT + N> M    4K + W(15) + 1M
.   .GOAL: FIND THE SECOND ROOT IN THE EXPRESSION
.   .   PRESS SHIFT + ALT + P> M  4K  + W(21) + 1M
.   .   PRESS SHIFT + ALT + N> M        4K + W(15) + 12M
.   .GOAL: CHECK THE VALUE OF THE INDEX  1M
TASK3 :20K + W(102) + 17M = 128
```

on ChromeVox -NavMatBR was 20 keystrokes, 15 mental operations and waiting for reading 77 words.

Listing 11: GOMS for Task 3 using ChromeVox-NavMatBR

```
GOAL: VERIFY THE INDEX OF THE SECOND ROOT OF EXPRESSION
.   .GOAL: FIND THE DESIRED EXPRESSION
.   .   PRESS SHIFT + ALT + N> M    4K  + w(30) + 1M
.   .   PRESS SHIFT + ALT + N> M    4K + w(21) + 1M
.   .   PRESS SHIFT + ALT + N> M    4K + w(15) + 1M
.   .GOAL: FIND THE SECOND ROOT IN THE EXPRESSION
.   .   PRESS SHIFT + ALT + N> R 4K + w(6) + 6M
.   .   PRESS SHIFT + ALT + N> R 4 K + w(5) + 5 M
.   .GOAL: CHECK THE VALUE OF THE INDEX 1M
TASK3 :20K + W(77) + 15M = 100.6
```

Listing 12 shows the GOMS model for Task 3 for the NVDA screen reader. The total number of operations on Task 3 was 7 keystrokes, 19 mental operations and waiting for reading 84 words.

Listing 12: GOMS for Task 3 using NVDA

```
GOAL: VERIFY THE INDEX OF THE SECOND ROOT OF EXPRESSION
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       PRESS DOWN ARROW       1K + W(34) + 1M
.       .       PRESS DOWN ARROW       1K + W(2) + 1M
.       .       PRESS DOWN ARROW       1K + W(2) + 1M
.       .       PRESS DOWN ARROW       1K + W(16) + 1M
.       .       PRESS DOWN ARROW       1K + W(14) + 1M
.       .GOAL: FIND THE SECOND ROOT IN THE EXPRESSION
.       .       PRESS UP ARROW   1K + W(2) + 1M
.       .       PRESS DOWN ARROW 1K + W(14) + 12M
.       . GOAL: CHECK THE VALUE OF THE INDEX 1M
TASK3 :7K + W(84) + 19M = 108.55
```

Listing 13 shows the GOMS model for Task 3 for the JAWS screen reader. The total number of operations on Task 3 was 12 keystrokes, 19 mental operations and waiting for reading 111 words.

Listing 13: GOMS for Task 3 using JAWS

```
GOAL: VERIFY THE INDEX OF THE SECOND ROOT OF EXPRESSION
.       . GOAL: FIND THE DESIRED EXPRESSION
.       .       PRESS DOWN ARROW       1K+W(2) + 1M
.       .       PRESS DOWN ARROW       1K + W(5) + 1M
.       .       PRESS DOWN ARROW       1K + W(2) + 1M
.       .       PRESS DOWN ARROW       1K + W(35) + 1M
.       .       PRESS DOWN ARROW       1K + W(2) + 1M
.       .       PRESS DOWN ARROW       1K + W(17) + 1M
.       .       PRESS DOWN ARROW       1K + W(2) + 1M
.       .       PRESS DOWN ARROW       1K + W(15) + 1M
.       .GOAL: FIND THE SECOND ROOT IN THE EXPRESSION
.       .       PRESS ENTER            1K + W(21) + 1M
.       .       PRESS RIGHT ARROW      1K+ W(1) + 1M
.       .       PRESS RIGHT ARROW      1K+ W(5) + 1M
.       .       PRESS RIGHT ARROW      1K+ W(4) + 1M
.       . GOAL: CHECK THE VALUE OF THE INDEX 1M
TASK3: 12K + W(111) + 13M = 129.96
```

Table 2: Estimated time to perform tasks on ChromeVox

Tasks	K	M	W	Time
1	16	25	73	107.48
2	12	21	87	111,36
3	20	17	102	128

Table 3: Estimated time to perform tasks on ChromeVox-NavMatBR

Tasks	K	M	W	Time
1	18	5	65	76.04
2	17	6	72	83.96
3	20	15	77	100.6

Table 4: Estimated time to perform tasks on JAWS

Tasks	K	M	W	Time
1	18	18	105	131.64
2	12	12	135	152.76
3	12	13	111	129.96

4.2 Time estimation comparisons between screen readers

As in a previous study performed by the research group [7], we used the values for time estimates suggested by Card [1] to obtain a quantitative number that could be used to compare the performance of screen readers analyzed in this article. Thus, the time for operator K (Keystroke) was assigned the value of 0.28 seconds for its execution. For the M (Mental) operator, the value of 1.2 seconds was used, and for the operator W (Wait) it was decided that the value of 1 second was used for each word, since this time is very relative to user preferences with their screen readers.

The time spent on each task was calculated given the following formula $Texecute = 0.28K + 1.2M + 1W$, the total time of each task on each screen reader can be found in Table 2, Table 3, Table 4 and Table 5 on column Time.

4.3 Discussion

As seen in the application of GOMS and KLM on the screen readers evaluated, quantitative values were provided, in order to perform a critical analysis of the proportion of effort and time blind users would have to devote in the execution of tasks of reading mathematical content in problem-solving tasks.

For this, we applied time values, widely used and advised in the academic scope [1], for each operator used in the models. This

Table 5: Estimated time to perform tasks on NVDA

Tasks	K	M	W	Time
1	6	22	72	100.08
2	3	19	73	96.64
3	7	19	84	108.55

Table 6: Estimated time for each reader to perform the tasks

Tasks	ChromeVox	ChromeVox -NavMatBR	NVDA	Jaws
1	107.48	76.04	100.08	131.64
2	111,36	83.96	96.64	152.76
3	128	100.6	108.55	129.96

yielded metrics that can be used to compare the performance of each screen reader evaluated. However, it is worth noting that these values do not reflect the times spent by people with visual disabilities, and that this value would probably be much higher.

It can be observed on Table 6 that the proof-of-concept prototype ChromeVox-NavMatBR had a decrease in the estimated time to perform the task, by providing the extra funcionalities, when compared to the original version.

The time to perform task 1 on ChromeVox-NavMatBR was 76.04 seconds against 107.48 of NVDA, 100.08 of ChromeVox and 131.64 of JAWS. This showed that the addition of features such as finding some specific term in the expression and using the part navigation to achieve the desired goal had a good effect in reducing the expected time and effort. On task 2, ChromeVox-NavMatBR demonstrated the same result obtained with task 1, showing that providing freedom to search for a specific element makes a significant difference.

Although JAWS allows the navigation in parts via the tree navigation, it still does not offer alternatives of search of specific elements. For example, on task 2, the lack of search of elements makes the expected effort required to find a specific element bigger than if it could be done directly. Another negative point found in JAWS is that the feedback messages to the user are very long, like in the case of activation of navigation by expression, which extends the time to read results.

Despite the limitations in the types of context explored in this proof-of-concept prototype, the results were very positive, and have pointed out directions to help design other strategies and to generalize this to other types of mathematical content.

5 CONCLUSIONS AND FUTURE WORK

With the results obtained by this study, we showed that there is a considerable potential for developing new ways of navigation in open source screen readers. This can make them capable of changing the way users with visual disabilities deal with mathematical content, since there are still gaps to improve the navigation capabilities of technologies that support mathematical content.

The results showed that enabling navigation using semantic markup of mathematical elements can enhance the expected user effort. This can lead to users being able to locate important content more easily as they solve mathematical problems read on the web, using similar navigations strategies as those available for different web content types, such as headings, paragraphs, links, and others.

Despite the limitations, the results from this paper are good indications for more research in the field, involving both users and Mathematics educators and experts. This could point out to new navigation strategies based on problem-solving approaches that consider the specificities of blind users.

At the moment, the research group has started a study involving users with visual disabilities to refine the current strategies and to help define broader schemes for the navigation. After this, user evaluations with the prototypes will be performed in order to obtain feedback from those who actually use screen readers in the mathematical context.

REFERENCES

[1] Stuart K. Card, Thomas P. Moran, and Allen Newell. 1980. The Keystroke-level Model for User Performance Time with Interactive Systems. *Commun. ACM* 23, 7 (July 1980), 396–410.
[2] Stuart K Card, Thomas P Moran, and Allen Newell. 1983. *The psychology of human-computer interaction.* L. Erlbaum Associates.
[3] David Carlisle, Patrick Ion, and Robert Miner. 2014. Mathematical Markup Language (MathML) Version 3.0 2nd Edition. (2014). Available online at https://www.w3.org/TR/MathML/, last accessed 15/01/2018.
[4] Alistair D. N. Edwards. 1989. Modelling Blind Users' Interactions with an Auditory Computer Interface. *International journal of man-Machine studies* 30, 5 (1989), 575–589.
[5] Alistair DN Edwards, Heather McCartney, and Flavio Fogarolo. 2006. Lambda:: a multimodal approach to making mathematics accessible to blind students. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility.* ACM, 48–54.
[6] Silvia Fajardo Flores and Dominique Archambault. 2012. Understanding algebraic manipulation: Analysis of the actions of sighted and non-sighted students. In *The International Workshop on Digitization and E-Inclusion in Mathematics and Science*, Vol. 2012.
[7] Anonymous for blind review. 2017. How Much Effort is Necessary for Blind Users to Read Web-based Mathematical Formulae?: A Comparison Using Task Models with Different Screen Readers. In *Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems (IHC 2017).* ACM, New York, NY, USA, Article 29, 10 pages. https://doi.org/10.1145/3160504.3160549
[8] Mick D Isaacson, Dave Schleppenbach, and Lyle Lloyd. 2014. Increasing STEM accessibility in students with print disabilities through MathSpeak. *Journal of Science Education for Students with Disabilities* 14, 1 (2014), 3.
[9] Martin Schrepp. 2006. On the efficiency of keyboard navigation in Web sites. *Universal Access in the Information Society* 5, 2 (2006), 180–188.
[10] P Schrepp, M. & Fischer. 2007. GOMS models to evaluate the efficiency of keyboard navigation in web units. *Eminds - International Journal of Human Computer Interaction* 1, 2 (2007), 33–46.
[11] Neil Soiffer. 2005. MathPlayer: web-based math accessibility. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility.* ACM, 204–205.
[12] Neil Soiffer. 2009. A flexible design for accessible spoken math. *Universal Access in Human-Computer Interaction. Applications and Services* (2009), 130–139.
[13] Volker Sorge, Charles Chen, TV Raman, and David Tseng. 2014. Towards making mathematics a first class citizen in general screen readers. In *Proceedings of the 11th Web for All Conference.* ACM, 40.
[14] Bernhard Stoeger, Mario Batusic, Klaus Miesenberger, and Philipp Haindl. 2006. Supporting blind students in navigation and manipulation of mathematical expressions: Basic requirements and strategies. *Computers Helping People with Special Needs* (2006), 1235–1242.
[15] Shari Trewin, Bonnie John, John T Richards, Calvin Swart, Jonathan P Brezin, Rachel K E Bellamy, and John C Thomas. 2010. Towards a tool for keystroke level modeling of skilled screen reading. In *ASSETS'10 - Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility.* 27–34.
[16] Web Accessibility in Mind (WebAIM). 2017. Screen Reader User Survey #7 Results. (2017). Available online at https://webaim.org/projects/screenreadersurvey7/, Last accessed on 17 March 2018.