Workshop on Advanced Learning Technologies for Disabled and Non-Disabled People (WALTD)
at
The 7th IEEE International Conference on Advanced Learning Technologies (ICALT 2007) July 18-20, 2007, Niigata, Japan

http://www.icaltd.doc.gold.ac.uk/

Mathematical working environments for the Blind: what is needed now?

Dominique Archambault[(1)], Bernhard Stöger[(2)], Mario Batu□ĩ[(2)],
Claudia Fahrengruber[(2)] and Klaus Miesenberger[(2)]

[(1)]INOVA/UFR922,
Université Pierre et Marie Curie, Paris, France
dominique.archambault@upmc.fr

[(2)]Integriert Studieren,
Johannes Kepler Univsersität, Linz, Austria
Bernhard.Stoeger@jku.at

Abstract

Blind people encounter great difficulties in dealing with Mathematics. Based on an analysis of these problems, we shall outline possible strategies to overcome them through software support. These strategies are currently being implemented in prototypes developed for a new Mathematical Working Environment dedicated to blind pupils and students.

1. Introduction

The study of Mathematics has always been particularly difficult for blind individuals. Indeed we can observe that a large majority of blind pupils do not succeed in Maths studies, while the average mainstream pupil succeeds more easily. As Maths is crucial in most science disciplines, this limits study options and future job opportunities for blind people.

We assert that there is no reason that Mathematical semantics can not be understood because of blindness; rather the biggest barrier is access to Mathematical content, which can only be through speech or Braille.

Most Mathematical concepts are best explained using drawings and notes which illustrate the main content (1).

(1)

In addition, the Mathematical notation itself uses 2 dimensions in order to convey the general structure of the formula rapidly, making the semantics easier to understand. One "pictures" the basic Mathematical content at a glance, which helps reading the details more efficiently, since the role of every part of the expression is already assimilated.

(2)

When visual modalities are not available, the situation is different. Other communication channels that are available to convey Mathematical contents (audio and tactile) do not allow a person to get a rapid overview. Indeed the representations used in both cases (Speech and Braille) are intrinsically linear, which means that formulas need to be linearised (3). In most cases this linearisation generates a much longer representation, which is more difficult to understand than the graphical one. For instance in this very simple example the linear version (3) necessitates 11 symbols while the graphical one (2) requires only 7.

$$(x+1)/(x-1) \quad (3)$$

The number of Braille symbols is also fairly limited: there are 6 dots available which can be combined into a maximum of 64 different patterns. Multiple Braille characters are therefore needed to code most Mathematical symbols. For instance, digits 1 to 9 are usually coded with the same Braille patterns as the first 9 letters. Prefixes then indicate whether it should be read as a digit, a lowercase Roman letter, an uppercase Roman letter, a Greek letter, etc...

To reduce the length of formulas, that is to reduce the number of characters and so facilitate understanding, Braille Mathematical notations use complex strategies. For instance, in the British code, the prefix is omitted in the case of a symbol which cannot be a number (after “j”), and lowercase Roman is assumed. In Marburg (German code), the prefix is used only the first time, like a switch, indicating that any other instance of the same pattern will be of the same type. In French and Nemeth (American code) the most frequent case is always assumed (lowercase Roman), and there is a prefix before each other (upper case, Greek, etc.). There is also a special way to represent digits: adding the dot ’6’ to the corresponding letter in French, or, in Nemeth, writing the same pattern on the lower part of the Braille cell (since those symbols do not use the 2 lower dots).

In the case of fractions, block markers identify the numerator and the denominator, making it necessary to reach the fraction symbol to determine that the expression is in fact a fraction. In Italian, the numerator and the denominator markers are not the same and there is no fraction symbol. Then when the first block marker is read, the user immediately knows it is a fraction. In the same kind of idea, Nemeth uses 3 Braille characters: the beginning of fraction, the fraction bar and the end of fraction.

To further complicate things, these Braille Mathematical notations have been developed in different countries, according to the linguistic and cultural history of these countries. Therefore, while the mainstream (visual) representation of formulas is identical in every language, the same is not true for Braille notations. Indeed each Braille Mathematical notation is widely used in its zone of linguistic influence, while it is completely unknown in other countries. In other words, a Braille formula written using the British notation is not understandable by a German speaking reader. This problem is quite important since the number of available Braille documents is very small compared to the number of ordinary Maths books.

2. State of the Art

During the last 2 decades a number of research projects proposed some partial solutions to this problem. First a list of projects focusing on access to Mathematical literature and preparation of Mathematical information have been developed. Most of these projects aim at converting mainstream formats to Mathematical Braille. These converters are used for different purposes. The main is to facilitate the production of scientific Braille documents. For instance they allow a teacher to use a document that was prepared for mainstream students and to convert it into Braille.

Labradoor [1] (LAtex to BRAille DOOR) converts a full LaTeX document including Mathematical formulas into Marburg Braille or into HRTEX (Human Readable TeX). Then several projects have been developed to convert MathML formulas: for instance Bramanet [2] produces French Mathematical Braille, math2braille [3] produces the Braille code in use in the Netherlands, and [4] produces Nemeth. As many Braille Mathematical codes exist, it seems interesting to create a programming library which is able to handle various Mathematical codes, as well mainstream as Braille, in a unified way, so applications using it could propose the best Maths code for each user. MMBT (Multi-Language Mathematical Braille Translator) [5] was a first attempt in this direction. It supported transcriptions from and to LaTeX, MathML, French (revisions 1971 and 2001), British and Italian Braille notations. MMBT has been developed in Java, it was discontinued and is now replaced by UMCL.

UMCL [6] (Universal Maths Conversion Library) is a programming library encapsulating various converters for different Braille codes in a single library, usable through a simple and unique API. UMCL is an open-source project and is portable. It was developed in standard "C" with wrappers to different other programming languages. To make this possible without increasing the complexity, an architecture based on a MathML as central representation of the formula was developed. Currently output modules have been developed for the French notations (revisions 1971 and 2001) and Italian. Beta versions of Marburg and British code are also already available. Input modules for LaTeX and Marburg Braille are also under development.

Another project which helps to produce documents usable by visually impaired people is Infty [7]. It is a large project aiming at giving access to printed Mathematical content. It is based on a core module (InftyReader) which is an OCR specialised in Mathematical documents. It produces a topological representation of the formulas in an XML format. Then this representation can be converted into various formats: MathML, LATEX, HTML, HRTEX, KAMS, and into Unified Braille Code (English) and Japanese Braille code.

In the other directions, converters allow sighted people to access documents written by blind people. This is more difficult to process since Braille codes are context sensitive. INSIGHT [8] proposes a complete system to translate Maths documents with mixed Grade II Braille text and Nemeth code to LATEX. The system processes an image of a Braille sheet (for instance a scanned page) and recognises the Braille dots to produce an ASCII Braille file. Text and Nemeth code are automatically identified and separated to be separately translated. Finally a single LATEX document is produced to be read by a sighted individual.

Several projects [9], [10], [11] focus on better presenting Mathematical contents in speech.

Now we need some new software tools that support the work of blind users, facilitating their understanding and helping them to carry out calculations, while facilitating inclusion in mainstream environment. Indeed more and more such pupils attend to mainstream schools, so it is necessary that these tools are usable with teachers who do not have a specific knowledge of Braille. A few projects have been carried out since a few years.

The Maths Genie [12] is a formula browser that facilitates understanding of formulas using voice. It has been designed to convey the structure of the Mathematical expression as well as its contents. The graphical rendering is synchronised to the audio. The current version supports English, French and Spanish for speech, and offers facilities to add any local language provided that a speech synthesiser is available with the requested interface. A Braille output supporting Nemeth code, based on [4], is available.

Lambda [13] is a Mathematical reading and writing system designed for blind students. The main characteristic of the Lambda project is that it is built on a brand new code. This code is an XML code specifically designed for supporting the Braille transcription into 8-dot pattern national codes. Each Lambda national code has the lambda structure and a Braille character dictionary as close as possible to the official national code. Lambda comes with a dedicated editor, which includes navigation support, input functions (keyboard shortcuts, menus, tool bar). It outputs Lambda Braille, speech synthesis (Mathematical symbols are verbalised in a descriptive language), a visual presentation in a linear code (a specific font in which each Braille character is represented by a visual symbol). A graphical rendering can be displayed on demand (but it is not synchronous to input).

3. Strategies to overcome the problem

The strategies presented here are implemented in MaWEn prototypes that are developed conjointly by Johannes Kepler Universität Linz and l'Université Pierre et Marie Curie within the framework of the MICOLE project. These prototypes are preliminary studies for the development of a Mathematical Working Environment dedicated for blind pupils and students.

3.1 To support collaborative work by synchronising views

One essential idea is that new tools should support collaborative work between blind and sighted individuals, most typically in a mainstream teaching environment, where a single blind pupil needs to be able to collaborate with a sighted teacher and possibly several sighted school mates.

This requires synchronisation of 2 representations using 2 different modalities, one for the blind and one for the sighted. Documents are composite: a main textual structured content includes Mathematical expressions. In MaWEn it is not yet possible to include other kinds of objects (like images with alternative content) but this possibility will be explored if the need appears from the users, for instance in the case of reading a schoolbook. Synchronisation of textual contents in both modalities is not particularly difficult, nevertheless care should be taken that the part which is displayed in Braille always appears on the screen and is clearly identifiable so that sighted users can easily follow the work of the blind pupil.

In the case of Mathematical expressions it is necessary that each of the representations synchronised needs to be a natural representation; that is the representation the readers are used to. In the case of sighted people it needs to be the natural graphical view, while In the case of Blind readers it has to be the Braille Mathematical notation they have been taught, that is the official Braille notation in use in their environment.

Synchronisation must allow each person to point at a location on the document, in the textual part as well as in a Mathematical expression, to show it to the other, in order to highlight an idea or to explain an error. With a graphical view this pointing should be done using the mouse by clicking on the desired location. The specified location would then be highlighted on the alternative view. In the other mode the Blind user must be able to make a selected location display with a different background on the screen.

The synchronisation must also support selection. The current selection must appear clearly as well on the screen as on the alternative display (Braille). This is achieved by showing the current selection with a different background colour. On the Braille bar it is possible to underline the selection using dots 7 and 8.

Synchronisation seems essential in inclusive education. Once again it is not a particular difficulty in the case of textual contents. For Mathematical expressions, it is made possible by the use of MathML conjointly with MathML to Braille converters. Indeed the MathML can easily be displayed graphically thanks to existing software. MathML to Braille converters enable the user to access in real time to a Braille transcription of the formula displayed on the screen. We have developed a model allowing to keep the Mathematical elements described in MathML and the corresponding Braille symbols linked. This model is implemented in the UMCL conversion library and allows MaWEn prototypes to support full synchronisation between the 2 views.

Remark: in the following sections we will only focus on the Mathematical expressions.

3.2 Collapse and Expand sub-branches of expressions

The main obstacle for a blind person to read and understand a Mathematical expression is the length of formulas and the complexity of notations. To get an idea about these crucial problems, and to explain the concept of collapse/expand which may help to overcome them, we would like to talk on the **structural tree** of a formula. As an example, let us consider this equation:

$$L_1 = L_0 .\ \overline{\boxed{1} - \frac{v^2}{c^2}} \quad (4)$$

It is an equation, with two sides: A “simple expression”, L_1 and a product, composed of another simple expression, L_0 , and a square root. The square root, in turn, is the difference between the number 1 and a fraction. The numerator of that fraction is v^2 , the denominator is c^2 .

This little discussion shows that our formula can be viewed as a tree; it is this very tree that makes up its Mathematical meaning, and that is understood by a sighted person at one glance, even if that person be not a Mathematician at all, or if that formula were even much more complex than this relatively simple example. On the other hand, the blind student will have considerable difficulty understanding the tree, and these difficulties are for sure more than proportional to the length of the formula. The blind person will have to do quite much reading forward and backward in order to finally understand this tree.

It is the idea of collapse/expand to assist the blind person in understanding the tree by presenting certain parts of it in full, while others are presented only in terms of blocks, informing the reader that more detailed information about the branch is available on request.

The above example formula (4) collapsed to the maximum possible, would be represented just by a block. Expanding this to one level, we get:

$$L_1 = L_0 . < \text{block} > \quad (5)$$

Expanding the block will finally give its content, namely:

$$L_1 = L_0 .\ \overline{\boxed{1}\square < \text{block} >} \quad (6)$$

When looking at the above tree expansions, it will be noticed that there are “jumps” in the expansion: For example, when expanding the right side of the equation, strictly following the tree structure would result in a block for the multiplication: $L_1 = < \text{block} >$, and then it would be necessary to expand this block to obtain the expression showed in (5). Also, expanding the square root should yield: $L_1 = L_0 .\ \overline{\boxed{<}\text{block} >}$. We did not propose to follow the structure in such a strict way because we think that this would make the process of expanding clumsy and, hence, less useful: sub-expressions of sufficient simplicity and brevity, like the left factor of the right side of our equation or the contents of the square root, should be displayed at once, without having to go through the strict expand process. It is considered an important task for the developer of a good collapse/expand algorithm to be “judicious enough” to keep these ergonomic aspects in mind.

This collapse/expand is not a new technique: It is used in “Integrated Development Environments” for quite a long time. It is implemented in projects to support blind persons in dealing with Mathematics, e.g., Maths Genie [12] and Lambda [13].

In the MaWEn prototype, we give headlines to the different blocks in order to make it easier to the user to understand the underlying structure. These headlines are mnemonic letters. To come back to our example, the formula, collapsed to the maximum possible, would be represented by the letter E, for “Equation”. Expanding this to one level, the square root is represented by the letter “Q”, so we would get:

$$L_1 = L_0 . Q \quad (5')$$

When expanding the Q, its content would be displayed using the letter “F” for fraction:

$$L_1 = L_0 .\ \overline{\boxed{1}\square < \text{block} >} \quad (6')$$

When looking at the above mnemonics like “E”, “Q”, “F” etc. one might ask how to distinguish them from ordinary Mathematical symbols. Indeed, a possibility of clearly distinguishing such mnemonics from

Mathematical characters must be found. As far as 6-dot Braille representations are concerned, this problem will not occur, because we shall use the mnemonics with dot 7. In case of an ASCII based Maths notation, however, other techniques need to be applied.

Another important requirement for collapse/expand support is the ability to synchronize it with the visual view. This means that, in a collaborative setting of a blind and a sighted person, a means must be found to make collapse of part of a formula visible to the sighted partner, typically, the teacher.

3.3 Editing functions

We do not consider it essential to use Braille Mathematical codes for inputting formulas. The main reason is that converters from Mathematical Braille to MathML are not yet available in the languages we need. It is also necessary to implement a solution which does not imply the use of a Braille keyboard, which is not available in every situation. Anyway as soon as such Braille to MathML converters will be efficiently available in UMCL, it will not be difficult to add this feature.

We propose instead a combined input scheme, where simple expressions such as numbers, variables, or polynomials can be input directly from the Braille or ASCII keyboard, while complex structures like roots, sub and superscripts, fractions etc. are to be input through commands (available in a menu and as keyboard shortcuts)

- **Numbers**: Ordinary Arabic numbers should be input directly, either through a Braille keyboard or through an ordinary qwerty keyboard. In the latter case, the number sign, which is common in many Braille codes, will be input automatically. As an alternative, input through a command should also be possible – this will prove helpful, e.g., in the rare case where Roman numbers are to be input through a standard keyboard. In any case, number input will be terminated by a space. Some caution will have to be applied when deleting or inserting digits: When a digit following the number sign is deleted, the number sign should be removed automatically provided there is no digit left. When inserting a digit it should be checked if another digit is present just after in order not to add a number sign.

- **Variables**: The same philosophy as for numbers will be applied. Special attention deserves the case where a variable consists of more than one letter. We can offer several strategies to handle this: In any case, we propose that letters input directly through the keyboard should be considered distinct one-letter variables, provided that they are not separated by spaces. Now in order to handle the case of multi-letter variables, one could either implement a command with a template (see below), or one might use spaces as separators, as is common with the well-known computer algebra system Mathematica.

- **Fences** (parentheses etc.): These symbols should be input directly through the keyboard.

- **Binary operators** (addition, multiplication etc.): The normal way of input should be via keyboard. However, commands may also be provided, creating templates like this: When the command for "Addition" is issued, an item like "placeholder + placeholder" might appear on the screen, where "placeholder" is a short symbol that can be easily recognized but still not easily confused with true Mathematical characters. For six-dot Braille codes, a double full six-dot cell would be an option. When an expression is input at the spot where the placeholder appears, then the placeholder will be replaced by that expression automatically.

- **Complex structures**: Complex structures such as sub and superscripts, roots, fractions etc. could be input through commands generating templates like in the previous example. An exception could be simple sub or superscripts, which might be input directly using the characters _ (underscore, for subscript) and ^ (caret for superscript), known from the TeX system.

To supply efficient mechanisms of selecting sub-expressions of a formula is very important, first because it is needed to realize collapse/expand support (see previous section), second, because a blind user needs to be able to select a portion of a formula in order to make it visible to his/her sighted partners.

The selection mechanisms are designed with the tree structure of an expression, as discussed in the previous section, in mind: This is because navigating through an expression will need that structure in any case. Pointing to a spot in a formula with the Braille display will select that node in the tree corresponding to the spot. Double clicking at a selected spot will select its parent node - through iterated application of that method, more and more of the expression, up to the whole one, may be selected.

There are some special cases to take care of: since not every character within a formula written in a Braille notation corresponds to a spot in its visual rendering, we need to provide for conventions on what should be selected when such a spot is pointed at on the Braille display. An example of such a character would be the opening and closing sign for a fraction, or the sign that announces the beginning of a sub or superscript. As a general rule, we propose that, when the user points at such a character, then the structure which is announced, or terminated, by that symbol will be selected. In particular, pointing at the "Begin of Fraction" or "End of Fraction" indicator will select the whole fraction, pointing at the symbol announcing a sub- or superscript will select the sub or superscript, pointing at the symbol announcing a root will select the contents of the root.

3.4 Manipulation support functions

Actually carrying out Mathematical calculations is even more difficult than reading and writing formulas. The problems in doing formal manipulations arise because of the complex structures that deploy during a calculation: sighted people may organise a computation such that it can be easily surveyed and navigated, and they use additional graphics to help their reasoning (see equation 1). Then there is a need for powerful editing tools that pupils can use to do calculations more easily. These tools should provide support with respect to the Braille notation itself and not to the Mathematical content.

We would like to develop our ideas on supporting blind persons in doing Mathematics by an example. Consider this exercise of multiplying two sums in parentheses:

$$(3a-5b+6c)\cdot(4a+b-3c) \quad (7)$$

The difficulties arising for a blind pupil already with such a relatively simple task were extensively analysed in [14]. Here, we shall describe how those difficulties might be reduced by implementing what we call a *Manipulation Wizard*.

To solve the exercise requires two steps: First the product must be expanded, meaning that every term of the left pair of parentheses has to be multiplied against every term of the right pair. Second, the resulting sum has to be simplified. We shall describe two wizards, one for expanding, and one for simplification, which in our example will be executed in sequence.

We begin with the expansion wizard: Every term of the left factor has to be multiplied by every term of the right one, the order being completely arbitrary. A wizard should address this by offering the various sub-multiplications to the student – (s)he is presented the factors, and needs just to enter the sub-products. By pressing the Down arrow key, the next sub-product should be automatically presented. For every sub-product, there should be an edit field to receive the calculated value from the user, e.g., in the line:

$$3a\cdot 4a=\ldots \quad (8)$$

There is an empty box after the equals sign, to receive the result $12a^2$ from the user. The wizard should collect all these results of sub-products, yielding a sum like this:

$$12a^2-20ab+24ac+3ab-5b^2+6bc-9ac+15bc-18c^2 \quad (9)$$

This sum is not yet the final result – it needs to be simplified. For this task, a *Simplification Wizard* should be designed: to simplify an algebraic sum, the following systematic procedure can be applied: Iterate through all the terms of the sum, choosing a "reference term" – this is the outer iteration. Compare the reference term to all terms coming after it, in order to find terms to be contracted – this is the inner iteration. If the reference term can be contracted with subsequent terms, collect the contractions for the resulting sum. If you do not find any successors of a reference term to be contracted with it, take the reference term into the result unchanged.

Even when strictly following this scheme, you may fall into two difficulties: First, you shall find terms succeeding a reference term that already were compared to a prior reference term, such that they do not need consideration anymore. Second, the procedure may let you view terms as reference terms which were successors of older reference terms, such that they also were already treated and are now to be skipped.

The *Simplify Wizard* addresses these difficulties by quite a simple procedure: As always, it presents you the various pairs "reference term – successor" beside each other. You may iterate through the successors with a pair of keys, e.g., Page-Up and Page-Down, leaving the reference term unchanged. You may take the next reference term with another key, e.g., the End key.

When a particular reference term is chosen, you may mark it with Control-M – this sets an opening square bracket left to it, which will remain adherent, such that, when you come across that term in the future, you will see that it was already considered. Also, it will be copied into a temporary edit buffer, allowing you to contract successors with it. Equally, when you decide a particular successor to be contracted to the reference term, then you should mark it with Control-M: This also makes an opening square bracket adhere to it, such that it can be seen as already having been used in the future; moreover, it copies the successor into the temporary buffer, to facilitate computing the contracted term.

Once a particular reference term has been fully processed, i.e., once all its successors have been considered, the sum representing the contraction, or the unchanged reference term, can be written into the temporary buffer. Once the various terms having formed the contracted sum were deleted manually, you may press the Down arrow key in order to copy the interim result into the final one, automatically clearing the temporary buffer to give room for the next reference term.

Although wizards of that kind would reduce much of the tremendous difficulties encountered by blind persons in doing Mathematics, it was argued that they are “too supportive”, which means that they take so much Mathematical work away from the pupil that (s)he might fail to understand the procedures of calculation. It is indeed an open research question how to develop software routines which, although being supportive, still leave enough responsibility on behalf of the pupil. As an example, the *Expansion Wizard* might be modified such that the pupil may carry out the iterations through the terms of the two factors by him/herself: By walking through one factor, (s(he may choose one of its terms as reference term by selecting it. In a second step, this reference term should be made constantly accessible to the pupil, while (s)he walks through the second factor in order to do the inner iteration by selecting terms of it. Once the result of a sub-product is computed, the student may issue a command to copy the result to a final sum, which (s)he may inspect after all the iterations are completed.

The ideal case would be to have a kind of “Method Base”, consisting of elementary support functions, from which more complex supportive wizards similar to the ones sketched here should be modelled through a configuration language, or, perhaps, through a wizard, in turn. Such a meta-tool should be made accessible to the teacher, or perhaps to a very advanced pupil, in order to tailor support to the pupil’s Mathematical maturity and abilities.

3.5 Context sensitive help for Braille codes

Additionally there is also a tremendous need for providing contextual support on the Braille Mathematical code. Braille Maths codes are normally hard to learn for a blind pupil, and almost impossible to learn for a sighted teacher. In order to support both the pupil and the interested teacher, a tool to furnish context-sensitive help for a Mathematical text written in a Braille notation would be desirable. Such a tool might display information about a character read by the Braille display, information about its meaning in Mathematical context for the blind partner, and, if possible, the black-print analogue of the character for the sighted partner. It should also offer concrete examples of its use, and an account of the important rules to be followed when using it.

4. Conclusion

We believe that the strategies described in this paper will effectively support the work of Blind individuals. They will be implemented in MaWEn and evaluated in school situations. One common thread with the other projects cited in the state of the art is the use of MathML. It is of tremendous importance to generalise the use of MathML in mainstream scientific documents so these documents can be easily accessed with those specialised tools.

Acknowledgements

The MaWEn prototypes are developed within the framework of the MICOLE project, which is funded by the INFSO DG of the European commission, within the framework of the IST 2002 programme (FP6/IST/eInclusion) under the reference IST-2003-511592 STP. This paper was written with the support of the @Science project which co-funded by the INFSO DG of the European Commission4, within the eContentPlus programme (FP6/thematic network priority), under the reference ECP-2005-CULT-038137. The contents of this paper is the sole responsibility of the authors and in no way represents the views of the European Commission or its services

References

[1] Batu□ĩ M., Miesenberger, K., and Stöger B. (1998), “Labradoor, a contribution to making Mathematics Accessible for the Blind”, In Edwards, A., Arato, A., and Zagler, W., editors, Proc. ICCHP 98 (6th International Conference on Computers Helping People with Special Needs), Oldenbourg, Wien, München.

[2] Schwebel, F. (2003). "BraMaNet logiciel de traduction des mathématiques en braille." [http://handy.univ-lyon1.fr/projets/bramanet/].

[3] Crombie, D., Lenoir, R., McKenzie, N., and Barker, A. (2004). "math2braille: Opening access to Mathematics." In Miesenberger, K., Klaus, J., Zagler, W., and Burger, D., editors, Proc. ICCHP 2004 (9th International Conference on Computers Helping People with Special Needs), volume 3118 of LNCS, pages 670-677, Berlin. Springer.

[4] Stanley, P. B. and Karshmer, A. I. (2006). "Translating MathML into Nemeth Braille Code." In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, Proc. ICCHP 2006 (10th International Conference on Computers Helping People with Special Needs), volume 4061 of LNCS, pages 1175-1182, Linz, Austria. Springer.

[5] Moço, V. and Archambault, D. (2003). "Automatic Translator for Mathematical Braille." In Stephanidis, C., editor, Universal Access in HCI - Inclusive Design in the Information Society, volume 4, pages 1335 - 1339, Mahwah, New Jersey, USA. Lea.

[6] Archambault, D., Fitzpatrick, D., Gupta, G., Karshmer, A., Miesenberger, K., and Pontelli, E. (2004). "Towards a Universal Maths Conversion Library." In Miesenberger, K., Klaus, J., Zagler, W., and Burger, D., editors, Proc. ICCHP 2004 (9th International Conference on Computers Helping People with Special Needs), volume 3118 of LNCS, pages 664 - 669, Berlin. Springer.

[7] Suzuki, M., Kanahori, T., Ohtake, N., and Yamaguchi, K. (2004). "An Integrated OCR Software for Mathematical Documents and Its Output with Accessibility." In Miesenberger, K., Klaus, J., Zagler, W., and Burger, D., editors, Proc. ICCHP 2004 (9th International Conference on Computers Helping People with Special Needs), volume 3118 of LNCS, pages 648-655, Berlin. Springer.

[8] Annamalai, A., Gopal, D., Gupta, G., Guo, H., and Karshmer, A. (2003). "INSIGHT: a Comprehensive System for Converting Braille based Mathematical Documents to Latex." In Stephanidis, C., editor, Universal Access in HCI - Inclusive Design in the Information Society, volume 4, pages 1245 - 1249, Mahwah, New Jersey, USA. Lea.

[9] Raman T.V. (1994), “Audio Systems for Technical Reading”, PhD thesis, Department of Computer Science, Cornell University, NY, USA.

[10] Stevens R.D. (1996), “Principles for the Design of Auditory Interfaces to Present Complex Information to Blind People”, PhD thesis, University of York, UK.

[11] Fitzpatrick D. (2006), “Mathematics: How and What to Speak”, In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, Proc. ICCHP 2006 (10th International Conference on Computers Helping People with Special Needs), volume 4061 of LNCS, pages 1199-1206, Linz, Austria. Springer.

[12] Karshmer A., Gupta G. and Gillan D. (2002), ”Architecting an Auditory Browser for Navigating Mathematical Expressions”, In K. Miesenberger, J. Klaus, and W. Zagler, editors, Proc. ICCHP 2002 (International Conference on Computers Helping People with Special Needs), volume 2398 of LNCS, pages 477-485, Linz, Austria. Springer.

[13] Schweikhardt, W., Bernareggi, C., Jessel, N., Encelle, B., and Gut, M. (2006). "LAMBDA: a European System to Access Mathematics with Braille and Audio Synthesis.", In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, Proc. ICCHP 2006 (10th International Conference on Computers Helping People with Special Needs), volume 4061 of LNCS, pages 1223-1230, Linz, Austria. Springer.

[14] Stöger B., Batu□ĩ M. and Miesenberger K. (2004), "Mathematical Working Environment for the Blind Motivation and Basic Ideas" In Miesenberger, K., Klaus, J., Zagler, W., and Burger, D., editors, Proc. ICCHP 2004 (9th International Conference on Computers Helping People with Special Needs), volume 3118 of LNCS, pages 656-663, Berlin. Springer.