

AudioMath

Towards Automatic Readings of Mathematical Expressions

Helder Ferreira

Diamantino Freitas

Faculdade de Engenharia da Universidade do Porto (DEEC) and
LSS – Laboratory of Signals and Systems
Rua Dr. Roberto Frias, s/nº, 4200-465 Porto, Portugal
dfreitas@fe.up.pt; helder.filipe@sapo.pt

Abstract

Thousands of technical and scientific documents are online in service web pages, personal homepages, e-learning websites and several other types of servers or databases. Mathematical expressions of all kinds can be found. Blind persons, for example, constitute a group of users for which the access to this information tends to decrease as the technical level of the document increases [1]. Some solutions were already tried for this problem, from Braille-based systems, to Text-to-Speech (TTS) engines.

The work presented in this paper focuses on the use of TTS technology to allow audio access to information for blind people, with relevant interest in the mathematical contents. It consists in the development of an accessibility tool – AudioMath (<http://lpf-esi.fe.up.pt/~audiomath>) [2] [3] [4] – capable to create automatic readings of mathematical MathML coded expressions (Mathematical Mark-up Language, <http://www.w3.org/Math/>). The rationale is that the formula's morphological structure is there to allow the creation of its textual description and the subsequent reading of the resulting text has to be done with the best possible cognitive effectiveness. How to read math expressions in terms of exclusively verbal description and in terms of prosodic cues like pauses and intonation is the main issue that is dealt in the paper and the use of an intra-formula navigation device is proposed for use in more complex expressions.

1 Introduction

Scientific and technical documents generally include many technical objects like formulae and charts or pictures that generally are difficult if not impossible to access if visual information is absent. This problem is especially critical for blind or partially sighted persons when they try to access this kind of material. The handicaps that have long been produced because of this difficulty are countless and therefore it is urgently required to contribute to alleviate this accessibility problem and remove the barriers posed to this part of the population in the access to complex technical contents of documents.

This is particularly pertinent as the opportunities for applying known basic solutions, like textual description of the contents, associated to audio rendering through synthetic speech, are increased by the recent appearance of de-facto standards such as XML-based mark-up languages, dedicated to more or less general coding of information of technical contents in structured documents. This is the case of MathML, but several other languages exist¹ and this is a field in steady development.

Two sectors must be considered in this respect of using mark-up languages to enhance the accessibility of technical or scientific complex documents: the production of the documents with MLs (mark-up languages) enabled authoring tools and/or special-purpose converters and the retrieval of the MLs-coded information existing in the documents by the appropriate user software, that normally has the functionality profile of a browser.

In particular, Web - based applications such as e-learning applications as well as browsers, must therefore be adapted in order to comply to the best known accessibility features based on use of structured documents by MLs.

¹ For example, SVG – Scalable Vector Graphics (<http://www.w3.org/Graphics/SVG/>), SMIL – Synchronized Multimedia Integration Language (<http://www.w3.org/AudioVideo/>), and others, like CML – Chemical mark-up language (<http://www.xml-cml.org/>).

In the field of engineering education, for example, the need for a continuous support for access to this type of documents for persons with partial or no vision is therefore crucial in order to remove related limitations in the access to this type of education.

Such support can be supplied by means of use of dedicated modules or plug-ins attached to browsers that introduce the functionality of performing some kind of medium conversion from the classic visual representation into another form like tactile information or audio/speech. The main goal of this paper is to describe the work done at LSS on the use of Text-to-Speech technology to provide the audio rendering of MathML.

We will present, in the second part of the paper an update of the state-of-the art in audio rendering of mathematical contents of documents that justifies the choice of the use of MathML and TTS for a wide range of applications. Subsequently, we will present the conceptual as well as the technical approaches that were selected in order to obtain the resulting audio rendering. In the fourth part results will be summarized to conclude and present following activities and some future possibilities.

2 State-of-the-art description concerning production and rendering of mathematical contents

The publication of scientific documents containing mathematical formulae is extremely demanding. The appearance of the TeX system, introduced in the late 70's by Donald Knuth (<http://www.tug.org/whatis.html>), solved the majority of problems with printed documents.

Other systems and trends existed since that time, like WYSIWYG (what you see is what you get) editors existed, until mark-up languages for Internet appeared, such as HTML (Hyper-text mark-up language).

HTML per se doesn't provide a mathematical description language directly embedded into the document. Therefore, other solutions were envisioned.

Combining HTML or XHTML (Extensible Hyper-Text Markup Language, that reproduces, subsets, and extends HTML, reformulated in XML) with Images (for instance, coded in jpg, gif, png (<http://www.libpng.org/pub/png/>), SVG...), Math expressions could be produced as images (raster or vector types) providing the visual rendering of mathematical expressions. The problem here is that this is a non-accessible format.

Other possibilities have been to use HTML + Symbol Fonts and (X)HTML + Cascading Style Sheets (CSS, a practical way of adding styles) that with the use tables, to geometrically structure information, achieves the desired visual rendering. As there is no embedded semantic meaning for the math expression, these are not accessible ways. One example is the translator TtH (the TeX to HTML translator).

Still another possibility has been the use of Applets to generate mathematical expressions. The result is slow and a non-accessible process. Example: WebEQ.

Word processors use proprietary or open formats for the documents, like Word / RTF / PDF / Postscript / TeX / LaTeX. When converting into HTML, the resulting documents produced from these materials still represent math expressions in the form of images (which are presently non-accessible). An example is the TeX4ht converter.

2.1 MathML

The creation and use of MathML [5] (Mathematical Markup Language - <http://www.w3.org/1999/07/REC-MathML-19990707/chapter1.html>) has brought out one of the most accessible solutions because it not only allows the visualization of math expressions on the web, but it also provides:

- Dynamic and interactive contents,
- The publishing of technical information in electronic format
- To swap math data between applications
- The interpretation of math expressions in non-visual media
- There are tools for conversion between LaTeX/TeX and MathML.

Other reasons for use of MathML exist, namely, the fact that it has been developed under W3C coordination, that it is gradually becoming a "standard" with a rapidly growing use by several relevant organizations associated with the teaching and learning of mathematical contents, as well as the involvement of software houses.

Additional aspects in favor of MathML are the emergence of editors and applications that create and manipulate

MathML documents together with the existence of conversion tools for the main publishing formats. Of course, MathML being an XML-based markup language facilitates its parsing, interpretation and conversion to other formats, and consequently a higher accessibility, portability and platform independence.

2.2 Main examples of systems dedicated to the audio rendering of mathematical contents of documents

One of the most impressive systems ever designed for this purpose was the ASTER (Audio System for Technical Readings). It is an application that accepts TeX notation as input and produces an ingenious audio rendering as output. It was developed by T.V. Raman in 1994 during his PhD. The system uses the Emacs front-end (Linux).

The ASTER has 3 main components, a Latex parser that creates an internal representation easier for the program to manipulate; the audio module that uses AFL (Audio Formatting Language) for rendering the parsed text using speech and other sounds and a browser that is used to help the audio rendering.

ASTER was a breakthrough. TV Raman's work is considered a 'bible' in mathematics audio rendering. The system supports a large number of mathematical formulae in LaTeX. Obvious limitations come from the inexistence of any kind of math formulae navigation support. It gets very complicated with complex math expressions to follow the audio output with effectiveness. However, ASTER uses a variable substitution process for resolving this problem with some success. Complex expressions can be divided in sub expressions at user's request.

Another system is MathPlayer, a mathematics display engine for Microsoft's Internet Explorer 6.0, developed by Design Science (<http://www.dessci.com/en/products/mathplayer/download.htm>). This system uses MathML Presentation markup as input and produces visual rendering (version 1.0 and 2.0) and audio rendering (only in version 2.0 – out in 2004) as output. There are a few shortcomings, related to the use of tables and the representation of matrices and the possibility of some ambiguous readings. There is no math formulae navigation support. Once again, it gets complicated with complex math expressions. Another aspect is that there is no provision for any kind of user adapted preferences scheme, that we could denominate "usermode" options.

2.3 Creating documents

Several authoring software packages allow a relatively easy composition and edition of mathematical expressions for inclusion in web documents:

- MathType, from DesignScience, an evolution of Microsoft Equation Editor, can produce documents in HTML + GIF or HTML + MathML formats. The MathPlayer plug-in is required for visualization. The code produced is MathML Presentation Markup [4].
- WebEQ, also from DesignScience, is composed of two packages, WebEQ Editor and WebEQ Publisher. They produce documents in the formats HTML + GIF, HTML + Java Applet or HTML + MathML. This code is either MathML Presentation Markup or MathML Content Markup [4].
- Mathematica and Publicon, from Wolfram Research, produce documents in the formats XHTML + MathML, XML (NotebookML), XML (NotebookML + MathML), TeX, HTML or TechExplorer. The resulting code is MathML Content Markup.
- Scientific Word, from Mackichan Software, produces documents in the formats HTML + MathML and LaTeX. The output code is MathML Presentation Markup.
- Amaya, from W3C, is a free browser with built-in editor producing documents in the formats HTML, XHTML, XML, Text, MathML, CSS and SVG. The output code is MathML Presentation Markup.
- EZMath, from David Ragget, is a free mathematical expressions editor with a special notation and browser plug-in. Output code blocks can be embedded in HTML and XHTML documents. The code is MathML Content Markup.

For TeX or LaTeX users there are tools like:

- Itex2mml that converts ITeX coded formulas into XHTML + MathML and the output code is MathML Presentation Markup.
- TeX2MML, an online converter from LaTeX to MathML Presentation Markup.
- TeX4ht converts LaTeX and TeX into HTML. By default produces GIF images for formulas, but can also produce HTML + MathML in the MathML Presentation Markup.
- TtM converts LaTeX or TeX documents in HTML + MathML and the code is MathML Presentation Markup.

- LaTeX2HTML converts LaTeX into HTML.

As can be seen, the MathML Presentation Markup dominates relatively to the MathML Content Markup, amongst authoring software.

3 Audiomath

In an initiative of the Laboratory of Speech Processing of FEUP, this project aims at building a tool (AudioMath) to operate either as a standalone application or integrated in a speech interface (TTS - text-to-speech) that does audio rendering of mathematical expressions according to the following scheme:

- Parsing, interpretation and conversion of MathML into plain text format
- Recognition and conversion of any text or markup elements not directly “understandable” as text by the TTS engine (numerals, abbreviations, etc)
- Generation of the appropriate prosodic contour for reading of the math formula’s text
- Use an intra-formula browsing device (navigation in the formula).

The rationale of Audiomath’s operation is that once the formula is described in MathML or equivalent code, the basic information available is enough to create the corresponding textual description as well as the reading of the resulting text with the best perceptual and communication effectiveness [6].

AudioMath has been designed as an ActiveX dynamic link library (dll) that can be used by any other program through internal calls.

Developed in Perl, the implementation was under Windows 9x/Me/2K/XP. However since it’s written in the Perl language, the main code can also be used under Linux/Unix.

The application is intended primarily for the reading of technical and scientific documents online in an accessible way, with particular benefit for vision-impaired persons.

Audiomath can also be used for teaching or learning how to read mathematical formulae as another area of application. Generally, in conjugation with a TTS engine, it can enhance the general accessibility to computer-based applications. In Figure 1, the main window of the application’s user interface is depicted.

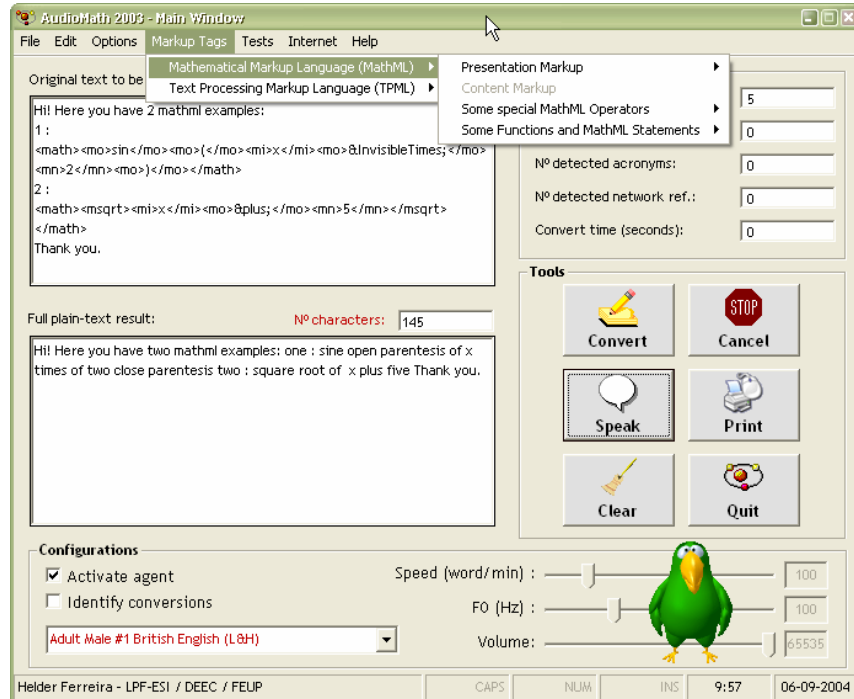


Figure 1: In this window the content of the first text box is converted and presented in the second. Some types of functionalities can be observed in the menus, buttons and boxes.

AudioMath has been built in a modular, extensible and configurable architecture. For the conversion part, it is organized in 6 major modules:

- Auto-Discovery (the “brain” of the operation that recognizes or identifies elements in the document and calls the respective conversion modules)
- Numerals (conversion of several types of numeric forms)
- Abbreviations (conversion of abbreviations)
- Acronyms (conversion of acronyms)
- Network References (conversion of IPs, emails and URI/URLs)
- Mathematical (conversion of MathML expressions)

A few examples of the several types of components of text in a document:

Acronym: UN – United Nations

Abbreviation: EQ. – equation; n° - number

Numeral: 1.2; +1,3; XV ;23° ; 1,333...

Web Reference: hfilipe@fe.up.pt

Math expression: $\langle \text{math} \rangle \langle \text{mi} \rangle x \langle \text{mi} \rangle \langle \text{mo} \rangle + \langle \text{mo} \rangle \langle \text{mn} \rangle 3 \langle \text{mn} \rangle \langle \text{math} \rangle$

Special Unicode character or a math glyph: é represents “é”.

The steps to follow in the conversion process are organized in the following sequence:

1. In the case of European Portuguese, convert all the Unicode into Latin1 code.
2. Launch the Auto-discovery process, based on regular expressions methods, that recognizes and classifies types of components (numerals, acronyms, ...)
3. Calls to the modules that convert the recognized elements into a full plaintext form.
4. Go to 2 and repeat until all types of components are converted.

To speed up the process the document should be divided into blocks of text, splitting the MathML markup from the rest of the text. In Figure 2 is depicted the flowchart representing the whole sequence of operation of the AudioMath system.

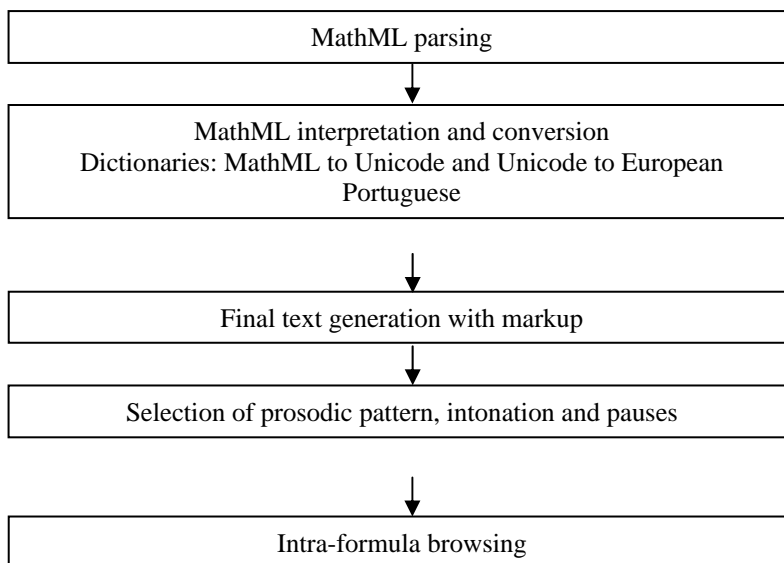


Figure 2 – The operation flow chart of the AudioMath system.

Text processing is strongly based on regular expressions and databases (for acronyms and abbreviations). For example, the following is a regular expression used for finding and converting percentage numbers:

```
if ($n =~ m/^(?:\+|\-|\+|\-)?[0-9]+(?:\.[0-9]+)?[\%]$/igs) { /* it's a percentage number*/ }
```

Dictionaries and databases were included inside the code as hash tables (allowing a higher run-time speed of execution but obviously less flexibility in doing updates of the contents).

The system supports usermode options. An example is the user preferring to hear the decimal part of decimal numbers spelled-out instead of spelled as a single number:

1.25 : one point twenty five or one point two five

In Figure 3, this specific characteristic of AudioMath for user adaptation by means of the so-called usermode configurations is observable. A number of features are represented in select menus and buttons for the case of conversion and rendering of numerals as an example.

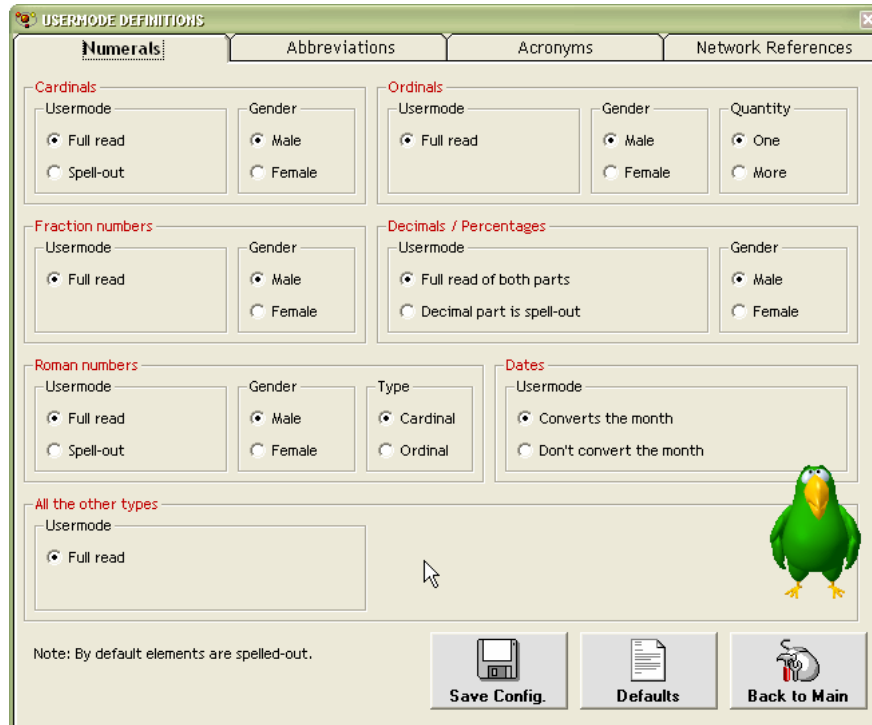


Figure 3: In this window the usermode options for the conversion of different types of numerals can be selected and saved.

MathML code is parsed using the XML::Parser module which acts as a SAX parser type (event-based), supporting encoding ISO-8859-1 (Latin-1) and discarding XML namespaces.

AudioMath works with MathML Presentation Markup so, a relatively bigger effort and computation is needed in the interpretation of the mathematical expressions.

Currently, in AudioMath, interpreting MathML is a process of using flags each time a starting and ending tag is detected, which allows to know the history of the sequence of the markup and to retrieve information, enabling to understand the structure of the math expression and do its conversion.

3.1 Database

As the expression is becoming discovered, the conversion process takes place by calling successive algorithms as well as Unicode and MathML dictionaries. AudioMath uses 2 kinds of dictionaries:

MathML entities to Unicode

Ex: $\⁢$ is converted to U+02062

Unicode to European Portuguese full plaintext form

Ex: U+02062 is converted into the Portuguese word *vezes* (it means times).

The conversion to full plaintext form is done according to a database of vocabulary and speech rules.

This database was designed by collecting the textual written form for each one of a comprehensive set of formulae.

Sub-sequent examination of the database allowed the extraction of the conversion rules from MathML to full-text for the more important mathematical structures appearing in the formulae of the database.

In Figure 4 an example of conversion can be seen in the main window of AudioMath user interface.

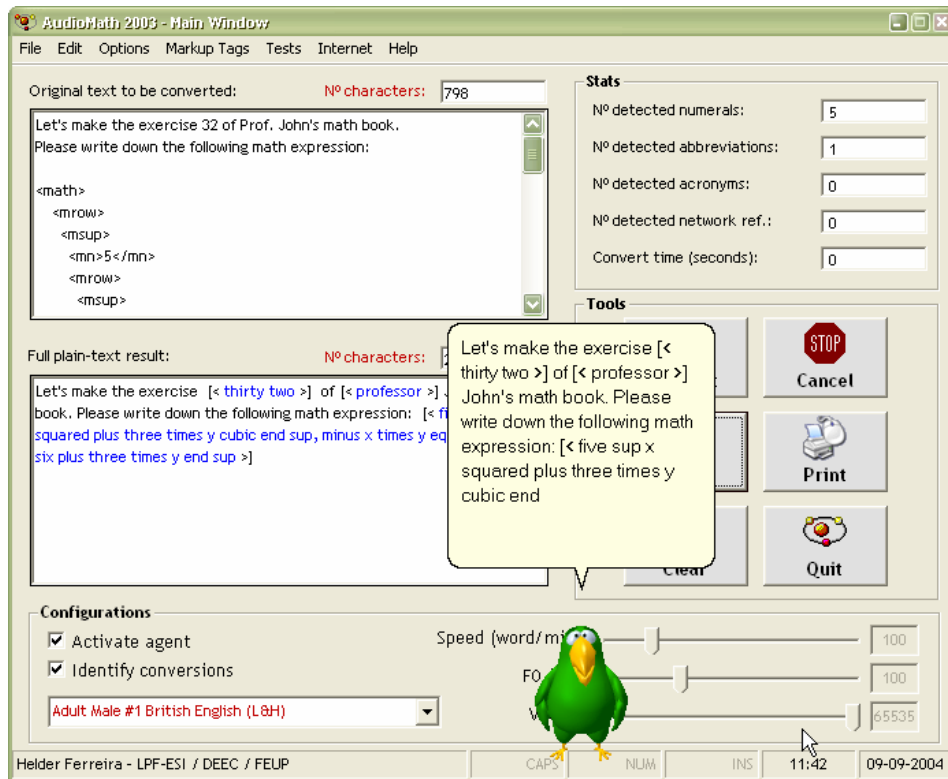


Figure 4 – This figure shows an example (in English for demonstration purposes) of the conversion from MathML to full-text.

It can be foreseen that the bigger the expression to convert is, the more complex the interpretation and conversion process becomes. Consequently the result will be a long text that takes correspondingly a long time to read by the TTS system.

As the hearing memory is not very efficient, the risk that the target listener not fully and accurately understands the meaning of the formula is not negligible. Therefore, in order to solve this problem an interactive navigational mechanism to browse the math expression should be provided in order to allow the user to browse the contents of the formula.

The structure of the browsing mechanism is organized in a tree-like way so that textual descriptions of elements follows an hierarchical order (either descending or ascending) and in this way allows an in-depth exploration of the formula elements by the user, according to her/his will and freely chosen sequence. Other schemes are also under investigation (work in progress).

This browsing must happen both externally to each element and internally (to infer the meaning of the mathematical contents).

3.2 Prosodic aspects

The objective of automatically speaking mathematical contents has to deal, besides the non-trivial issues of text generation and phrasing, with the generation of prosody to superimpose over the synthetic speech waveform [7].

For example, consider the two possible readings of the expression shown in Figure 5:

- 1 - Square root of a squared plus b squared, end of radicand.

2 - Square root of power base a exponent two, end of power, plus power base b exponent two, end of power, end of radicand

Which of these forms is more correct, not ambiguous and more efficient?

$$\sqrt{a^2 + b^2}$$

Figure 5 – A simple example formula used to illustrate the need for a carefully selected prosodic contour.

An interesting experience is to read a math expression monotonically to someone that is not looking at it and ask for a written version after the dictation. The result is that if one is not careful enough when reading the expression, like in example number 2 (above) that also brings a non-negligible overhead in text length, different formulas may all sound much alike and therefore identification and understanding of the meaning or semantics of the formula will be difficult. The monotonous reading introduces ambiguity in the understanding of the formulae.

The solution in the opinion of the authors must pass by the adoption of formal ways of text generation that keep the right structure information of the formula, combined with a selection of the right prosodic structure (f0 contour, pauses, ...) to superimpose over the speech signal so introducing the right pauses and intonation patterns.

One of the AudioMath's current tasks is the continuous update of the database of vocabulary and speech rules, for a growing number of subsets of mathematical formulae.

These speech rules as well as the pauses and intonation patterns are implemented at conversion time by tagging the text with custom prosodic marks, that are subsequently used to command the TTS engine in order to produce the required values and time positions of pauses and f0 modulations. In the near future the prosodic mark-up associated with the current TTS engine will be converted into one of the existing standard languages.

The Math corpus is defined and categorized by different operation types: basic algebra, trigonometry, integral and differential calculus, etc.

Each type has an analyzed structure and is defined in a formal plaintext way.

Different readings of the same formula were spoken and analyzed for prosody and perceivness. Pitch patterns and pauses were detected and their values extracted from speech for later use in the prosodic mark-up in the form of rules. This information constitutes a specific database of prosodic information.

For illustration of the importance of the prosodic patterns that the formulae readers use in order to help the listeners to understand the meaning of the formulae, an example is given in the following. The formula of figure 5 was read according to the textual version number 1 above. The speech waveform was analyzed using the Praat software and f0 contour as well as pauses were detected and measured as can be seen in . These prosodic events (short or long pauses, onset and offset of each f0 modulation) were subjectively time correlated with the boundaries of the text segments related to each of the formula elements, looking for identification of prosodic rules. Some prosodic conclusions were then inferred from this correlation between texts and waveforms. To start with the pauses, in this case there are 2 distinct types: long pause (LP) and short pause (SP) occurring in completely different boundaries of the textual form, as can be seen in the following:

Raiz quadrada de (LP) á ao quadrado (SP) mais bê ao quadrado (LP) fim de radicando.
(Square root of (LP) a squared (SP) plus b squared (LP) end of radicand)

The occurrence of the short pause is optional on the contrary of the long pause that is mandatory.

Next, the f0 contour reveals explicitly relevant rising and falling movements of f0 in the intonation that the speaker's used with careful synchronism with the pauses, in the intention to provide cues for the listener to be able to recover the hierarchical classification of the boundaries introduced by the pauses. The observations point out the following rules:

- 1- Rising tone: used when a lower hierarchical level is starting. (root of...)
- 2- Falling and Rising tone: used to mark the smaller separating pause. (...a squared...)

- 3- Falling tone: used when level is ended. (...b squared...)
- 4- Emphatic Falling tone: used at the end of the expression that simultaneously is the end of the higher hierarchical level (...end of radicand).

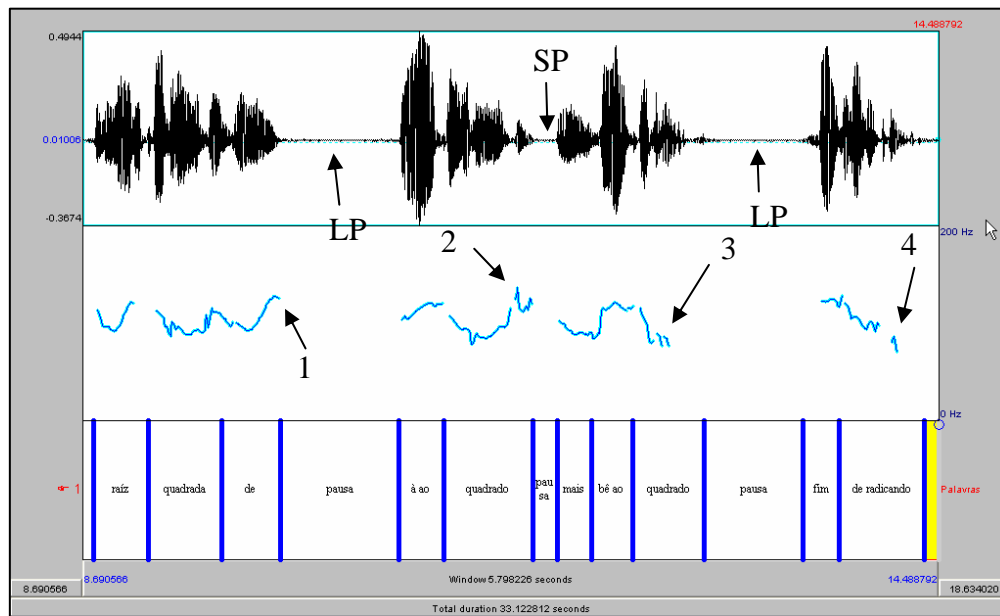


Figure 6 - Analysis of the utterance corresponding to the formula of figure 5, spoken in Portuguese.

4 Conclusions and future work

In this paper a new system for audio rendering of mathematical formulae was presented. It is intended to be an accessibility tool, especially important for persons with partial or total loss of sight to be able to access mathematical contents, in the case these are coded in MathML. Taking in consideration the state-of-the-art of the rendering of mathematical formulae in documents, the conclusion was that the use of MathML is the right option, because it is an official language (W3C) and it makes possible to code most mathematical formulae.

The presentation and the content varieties of MathML were considered and, even if the content variety is more desirable, due to its proximity with the semantics of the formula, easing the task of interpretation of the code, the presentation variety was selected because of the current availability of authoring software for documents almost all in MathML Presentation Mark-up.

The development of the current version of AudioMath followed a module-based structure in order to cover the operations necessary to convert the formula into prosodically rich speech.

Starting by analyzing, interpreting and converting the MathML contents according to a set of rules extracted from a specifically designed database, the presence of lower level elements like numerals and abbreviations, etc, is subsequently detected in the text and a conversion to full-text format done by specific modules takes place.

The analysis of the structure of the formula that was extracted from the MathML code is then used for assigning the right prosodic patterns to the text in order to produce a correct reading of the formula, according to another set of prosodic rules that were investigated by means of a specifically built database of read utterances corresponding to the formulae database and their corresponding textual forms.

The final form of the text with prosodic mark-up is then supplied to the TTS engine for the speech output to be produced.

The user is assisted in the listening process by a navigational interface that allows the user to make back and forth movements of the reading point of the formula in order to increase understanding as much as possible. These movements are organized in a tree-like way with the structure of the specific formula.

Currently, the system is capable of operating automatically for a large number of formulae varieties and the number of types in continuously being increased through continuous development work.

The system has full Unicode support (MathML and Unicode Dictionary for European Portuguese) and the MathML Presentation Markup tags fully supported are: (Token elements) `<mo>`, `<mi>`, `<mn>`, `<mtext>`, `<mspace>`, `<ms>`, `<mglyph>`, (General Layout Schemata) `<mrow>`, `<msqrt>`, `<mroot>`, `<mfrac>`, `<mstyle>`, `<merror>`, `<mphantom>`, `<mpadded>`, (Tables and Matrices) `<mtable>`, `<mtr>`, `<mtd>`, (Enlivening Expressions) `<maction>`, `<none>`. The MathML Presentation Markup tags still partially supported are: `<msup>`, `<mfenced>`, `<menclase>`, `<mtable>`, `<mtr>`, `<mtd>`

The auto-discovery device can detect and convert into full-text numerals (cardinals, ordinals, decimals, romans, percentages, dates, time, scales, sport results, fractions, currency, powers, telephones and engineering formats), abbreviations (social, currency, chemistry and physics (on database)), acronyms (several (on database)), Network references (email addresses, url/uri and ips). A browser was built for the functionality test and TTS integration. The system has usermode support of preferences on rendering. Basic studies on mathematical prosody and speech database rules for mathematics were done with satisfactory results, although more development is needed in order to integrate with the intra-formula navigational system.

Informal user evaluation has been used in the several iterations of the AudioMath's development. A new evaluation phase for the effectiveness of understanding of the spoken formulae is about to start at edition time and results are expected before the time of presentation of this paper.

4.1 Future Work:

To complete the support to the MathML Presentation Markup and add support to the MathML Content Markup are some of the most important tasks to be executed.

Anyhow further learning on how to read mathematical formulae will be reached through continuous development and testing of the formulae database.

Another aspect that deserves attention is to widen the coverage of the available codes, both for the documents formulae and to the TTS engines, for example, supporting modules for HTML, XHTML, SSML and others.

Continuously testing and improving the mechanisms for navigating inside the mathematical formulae (eventually building a specialized audio browser) is also a needed activity.

Adding support for new languages (English, French, German, etc.) and developing further the study on the prosody of reading the mathematical formulae are also activities that will get support in the near future.

5 References

- [1] Fitzpatrick, Donal. Monaghan, Alex. Browsing Technical Documents: Document Modeling and User Interface Design. BULAG 1999 Proceedings.
- [2] Ferreira, Helder. Freitas, Diamantino. Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project. ICCHP 2004 Proceedings. Paris, France.
- [3] Ferreira, Helder. Freitas, Diamantino. Audio Rendering of Mathematical Formulae using MathML and AudioMath. UI4ALL 2004 Proceedings. Vienna, Austria.
- [4] Ferreira, Helder. AudioMath: Speaking Mathematics with MathML. 2nd European Workshop on MathML and Scientific e-Contents 2004. Kuopio, Finland.
- [5] W3C Recommendation. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). <http://www.w3.org/TR/MathML2/>.
- [6] Karshmer, Arthur. How Well Can We Read Equations to Blind Mathematic Students? HCII2003 Proceedings. Volume 4. Crete, Greece.
- [7] Stevens, Robert. Principles for the Design of Auditory Interfaces to Present Complex Information to Blind People. PhD Thesis. 1996.