Improving Automatic Speech Recognition for Mobile Learning of Mathematics Through Incremental Parsing

Marina ISAAC[a], Eckhard PFLUEGEL[a], Gordon HUNTER[a],
James DENHOLM-PRICE[a], Dilaksha ATTANAYAKE[a] and Guillaume COTER[b]
[a] *Kingston University, London, U.K.*
[b] *Ecole des Mines de Douai, Douai, France*

Abstract. Knowledge of at least elementary mathematics is essential for success in many areas of study and a large number of careers. However, many people find this subject difficult, partly due to specialised language and notation. These problems tend to be even more marked for people with disabilities, including blindness or impaired vision, and people with limited motor control or use of their hands.

Learning activities are increasingly carried out in mobile computing environments, replacing the roles of textbooks or handwritten lecture notes. In the near future we expect the modality of use of mobile devices to shift from predominantly output to both input and output. The generally poor performance and usability of text input on these devices motivates the quest for alternative input technologies. One viable alternative input modality is automatic speech recognition, which has matured significantly over the last decade.

We discuss tools that allow the creation and modification of mathematical content using speech, which could be of particular benefit to the groups of disabled people mentioned above. We review previously proposed systems, including our own software for use on desktop PCs, and conclude that there is currently a lack of tools implementing these requirements to a satisfactory level.

We address some technical challenges that need to be solved, including the problem of defining and parsing of suitable languages for spoken mathematics, and focus on the parsing of structured mathematical input, highlighting the importance of incrementality in the process. We believe our incremental parsing algorithm for expressions in operator precedence grammar languages will improve upon previous algorithms for such tasks in terms of performance.

Keywords. Learning mathematics, mobile learning, disabled students, Automatic Speech Recognition, parsing

Introduction

Learning is increasingly enhanced by technology, and the surge in use of mobile devices such as Tablet PCs and smart phones offers new learning opportunities [1]. Students benefiting from mobile learning technology include, for example, online (distance) learners, people working or studying "on the move", and sometimes disadvantaged users such as people with physical disabilities or other special needs [2].

Until relatively recently, automatic speech recognition was only available to a small group of highly specialised scientists. Even around twenty five years ago, in

order to use what was then considered state-of-the-art speech recognition software, highly expensive equipment was required. Today, even hand-held devices such as iOS or Android smartphones, tablet PCs and game consoles such as the Xbox and Nintendo Wii are often equipped with some form of speech recognition software. The reasons behind this rapid evolution are the tremendous progress that has been made in speech recognition technology, the advances in available memory and processing power of personal computers and the fact that complex and computationally intensive tasks can now be carried out with the help of powerful cloud-based applications. At present, speech recognition is used in a large number of application domains. As a consequence, speech input plays an increasingly important role in making computer tasks accessible to users who wish or need to rely on input modalities other than conventional keyboard and mouse. However, despite its great importance within the curriculum, both at school level and in higher education, for specialists and non-specialists (including students of Science, Engineering, Economics and Business subjects), the development of such technology for making Mathematics more accessible both to disabled and other users lags a long way behind its use in more conventionally alphabetic text-based subject disciplines.

One of the main motivations behind our work is to make speech-driven systems as suitable as possible for users who typically might wish to engage in hands-free computing, and in particular to facilitate mobile learning of Mathematics. This is realised through our TalkMaths project [3] which has delivered several prototype systems and interfaces, as described in Section 2, the latest under development being a mobile app.

The main contribution of this paper is, as part of a progress report of recent activities within the TalkMaths project, a high-level description of an efficient implementation of incremental processing of speech-input, commonly referred to as incremental parsing. Our algorithm, an improvement of an existing algorithm by Heeman, consumes less battery power and provides a more responsive interface, and is hence suitable for a mobile version of the existing TalkMaths system.

This paper is organised as follows: in Section 1, after briefly reviewing existing commercial and freely available automatic speech recognition tools, we discuss past work on speech-driven editors for mathematics and the challenge of defining spoken mathematics. In Section 2, we present the current architecture of the TalkMaths system and the used parsing technique. Section 3 describes our main contribution of this paper. We conclude our findings and propose future developments in Section 4.

1. Interfacing Mathematics using Speech Technology

In this section, we discuss issues that arise when creating mathematical content using speech as input. We review the state of the art of automatic speech recognition technology and discuss approaches to formalised spoken mathematics, including more in detail the language used in the TalkMaths system.

1.1. Review of Speech Recognition Technologies

Automatic speech recognition (ASR) is the process of converting human spoken words to human- or computer-readable text [4]. Most ASR systems are speaker dependent –

i.e. they require to be trained by a particular user in order to provide the best and most accurate recognition of that user's speech.

There are several commercial ASR packages available in the current market, with the most widely available ones being sold by Nuance and Microsoft. With claims of 99% word accuracy under good conditions, Nuance Dragon NaturallySpeaking (DNS) [5] is the market leader. Originally developed for Windows, recent versions of DNS also run on Mac OS X. Microsoft has included Windows Speech in Windows 7 onwards, and their speech recognition solution is likely to be further improved in future versions of their operating systems. At present, it remains unclear which of these two main players will eventually dominate the market.

A free alternative to the commercial products is Sphinx, an Open Source toolkit for speech recognition [6] developed at Carnegie Mellon University. With Sphinx, additional data such as prosodic and intonation information can be captured. Currently, the recognition accuracy is inferior to that of the commercial ASRs.

Increasingly, vendors are adding speech support to their software. For example, Google has enabled speech recognition in its Chrome web browser, specifically designed for speech-enabled applications, such as "Google Voice Search" [7]. In particular, mobile devices and gaming consoles are widely speech-enabled, with speech controlled assistants being offered by Google and Apple.

Complementary to ASR technology is speech synthesis (or Text To Speech, TTS) technology, whereby a computer system tries to create a comprehensible set of sounds approximating the way a person would read a particular word, or sequence or words, aloud. However, detailed discussion of TTS technology is beyond the scope of this paper, and we refer the reader to [8] for further details.

1.2. Previous Work and Overview of Existing Systems

We are aware of a number of systems translating spoken mathematical input to different output formats and displaying the structure of the mathematical expressions. These work together with ASR systems installed on the user's machine.

MathTalk [9] is a commercially available system that implements speech macros for use within the Scientific Notebook environment. Its functionality, even when compared with the other academic prototype systems mentioned below, is quite limited.

Fateman has undertaken work leading to Math Speak & Write [20], a multimodal approach combining spoken input with handwriting recognition. Unfortunately, the ambitious aim of simultaneous multimodal input was not achieved. Another system is CamMath, described in [10], which needs the same support environment as MathTalk but seems to offer a better developed command language.

Previous approaches to allowing spoken input of mathematics include the research prototype systems of Bernareggi & Brigatti [11] (which only works in Italian) and Hanakovič and Nagy [12] (restricted to use with the Opera web browser due to it using X+V (XML + voice) technology).

While various mobile apps exist that parse mathematical input, entered either via an on-screen keyboard or by drawing symbols on the screen, none offer dictation as an input modality. Although speech to text input is widely available, its functionality conforms to standard ASR offerings, and does not handle specialised languages such as those needed to dictate mathematics or computer program code.

We conclude that all these systems are not yet robust enough for day-to-day use by (potentially inexperienced) users.

Our work on the TalkMaths system effectively touches all these aspects. TalkMaths is a web-based structure editor, which accepts natural language commands to control the editing process. We will assume that spoken mathematics commands have been recognised by the ASR system, or perhaps have been directly typed into a suitable user interface. As a consequence, we have a string of *transcribed spoken mathematics* which we would like to further analyse with the ultimate goal of displaying it on-screen or on a printed page. Each successive prototype version of TalkMaths achieves this, within the constraints of their respective limitations. TalkMaths has also been employed in a pilot study to investigate whether its use could also enhance students' learning of mathematical terminology and concepts [13].

1.3. Defining Spoken Mathematics

A somewhat specialised language is required to describe mathematical expressions, imposing a compromise between removing ambiguity and making the language too complicated to use. This is covered by the work of Chang [14], Raman [15], Wigmore [16] and Attanayake [17].

Defining a standard for spoken mathematics is an essential starting point when creating speech-based mathematical editing systems. In this section, we will review several approaches that have been taken for this.

The question of how to define rules for speaking (or reading aloud) mathematics has been posed by several authors in the past. The motivation behind these attempts to find standards seems to be linked to three different contexts: dictating or describing mathematics to other human beings, parsing mathematics by computer systems to further process the input, and defining rules for converting mathematical content to audio synthetic speech output (text-to-speech, TTS).

One of the first attempts to give a standard for speaking mathematical equations and objects in English appears to be Chang's handbook "Larry's Speakeasy" [14]. He defines spoken forms for a broad range of mathematics, from basic symbols, algebra, trigonometry, logic, geometry, statistics, calculus, linear algebra, topology to mathematical diagrams and graphs. Chang, himself being a blind mathematician, primarily focuses on dictating mathematics to other human beings.

As far as we are aware, the most complete investigation into spoken mathematics to date is presented by Fateman [18, 19]. This focuses on introducing a vocabulary that is intuitive and easy to learn by novices, and at the same time allows as little ambiguity as possible. This work was carried out within the context of the Math Speak & Write system [20]. Fateman provides a detailed analysis of how to speak numbers, non-numeric tokens, nested arithmetic expressions, integrals and sums. He also discusses the problem of ambiguity, to which in some cases no easy solution seems to be available.

Apparently unaware of this work, Elliott and Blimes [10] developed a similar approach, although their language design is oriented towards the use of "two-dimensional mathematics" (such as expressions describing integrals) in combination with an existing mathematics editor (*Scientific Notebook*).

In [16, 21], spoken mathematics is also investigated, based on empirical evidence of how people – notably mathematics students and teachers – actually speak mathematical expressions. This was carried out by analysing transcriptions of recorded mathematics classes from the British National Corpus (BNC) [22] and an experiment where participants read out given expressions. The study focusses on the potential of

prosodic information providing clues to resolve ambiguity. However, this did not influence the design of the language used in older versions of TalkMaths, and Wigmore [16] uses an approach very similar to that of Fateman.

Rules for spoken mathematics have also been developed for text-to-speech conversion of mathematics, which can be of particular interest to blind and visually impaired students. For example, Raman, another blind mathematician, gave a framework in his ASTeR system [15], which gave synthetic readings of LaTeX/TeX documents including describing equations and formulae in words for blind users. A rather more recent system for the same purpose is STEMReader [23], based on MathML rather than LaTeX. Another initiative is the MathSpeak project [24] in which a set of grammar rules for speaking mathematics have been designed. However, rules for speaking mathematics that have been developed for text-to-speech appear less suitable for use with spoken input due to the tedious learning curve and usage they require. In other words, a text-to-speech system would use more descriptive language, probably employing more words, to give the listener as much information as possible to describe and explain the expression on screen, which might be too long, verbose and complex for dictating and inserting relatively simple formulae [19]. One approach for evaluating such systems is EAR-Math, developed by Kacorri et al [25].

It should be mentioned that some commercial systems are gradually starting to support spoken mathematics. The most commonly-known example is the Computer Algebra System *Mathematica*, that actually provides a function spokenString() [26] for converting mathematical objects into their spoken language form.

1.4. Spoken Mathematics as Input Language

In this section, we present the main characteristics of the language for spoken mathematics that we use in the TalkMaths system, for which we a parser has been developed [17, 27] that includes error recovery strategies.

Our language consists of a set of spoken forms for operators and operands. This approach is not new and, arguably, most previous attempts follow the same scheme. However, we improve several aspects of spoken mathematics, as explained in the remainder of this section. We note that writing LaTeX or MathML is not convenient or suitable for novices such as school pupils, or some disabled students.

1.4.1. Ambiguity

As already stated, and noted in the literature [16, 19], spoken descriptions of mathematical expressions often contain ambiguity. Precedence rules, normally expected to be known by mathematically trained users, can help with resolving this ambiguity. For example, the phrase "alpha plus bravo over charlie"[1] could mean either $a + \frac{b}{c}$ or $\frac{a+b}{c}$. If a display of the expression were available – perhaps in a teaching session or during a mathematical talk – there would be no confusion.[2] Otherwise, if we need to interpret the transcribed spoken mathematics without additional clues, our system should follow the rule that division (the fraction operator) binds more strongly

[1] The NATO alphabet is used to avoid confusion between similar sounding common English names of various letters, such as "b" ("bee"), "p" ("pee") and "v" ("vee") [19].

[2] In the case of blind or severely visually impaired people, an appropriate spoken version, or tactile rendering (e.g. by Braille), of the mathematics would need to be delivered.

than addition. Hence, following this precedence rule, we would decide on the first expression rather than the second. However, it should be noted that people may not always follow these rules precisely when speaking mathematics!

In other cases there is not always an obvious solution. Consider the transcribed spoken mathematics, "square root of bravo squared minus alpha". This could either be interpreted as $\sqrt{b^2} - a$, or $\sqrt{b^2 - a}$. No obvious operator precedence rule would help in this example. There is no common standard that indicates exactly how much input relates to the argument of the square root function.

The approach taken by Fateman [19] is to use additional "locutions" (spoken delimiter/marker commands) to clearly indicate boundaries between arguments. The first expression would be spoken as "square root of bravo squared all minus alpha". Here, the term "all" acts as an "invisible" end marker to separate the square root function from the subtraction operation. An additional construction ("quantity") implements a corresponding "begin" marker. The expression $a + \frac{b+c}{d}$ for example would be dictated as "alpha plus quantity bravo plus charlie all over delta". Note that these commands could also be used as a spoken form for brackets, as for example in the input "alpha plus bravo all times charlie" which is the spoken form of the expression $(a + b) \times c$.

The emphasis of the basic design of spoken mathematics in [19] is on simplicity and convenience for the user. For these reasons, the author of [19] does not consider the use of additional commands other than "quantity" and "all".

1.4.2. Speech Templates

Attanayake [17] extends this scheme to include the use of *speech templates*[3]. Speech templates are (typically, relatively short) spoken natural language commands. A speech template consists of one or several groups of words. Between each group, any other spoken language elements (including speech templates) might occur. Effectively, these groups act as "boundaries". Hence, a speech template can take zero or more arguments and can be nested.

For example, consider the expression $\left(a + \frac{b}{c+d}\right) \times e$ (see Section 3.6 in [19]). Using Fateman's approach we would say "alpha plus bravo over quantity charlie plus delta all all times echo". Suppose we define a speech template for fractions as follows.

```
"fraction" .. "over" .. "end fraction"
```

Using this, the expression is now pronounced as "alpha plus fraction bravo over charlie plus delta end fraction all times echo". This appears more natural, as it avoids the need for the repeated "all all" construction.

Speech templates without any arguments can also make sense. These consist of a single command such as "edit expression" or "what can I say". Such commands can be useful, for example, in a system that offers additional functionality such as editing mathematical expressions or displaying help information.

An example of a speech template with more than two arguments could be for speaking or typing definite integrals, as:

```
"integral from" .. "to" .. "of" .. "end integral"
```

[3] These should not to be confused with the type of acoustic templates used for pattern matching in speech signal processing.

The expression $\int_{x=a}^{b}\big(f(x)+g(x)\big)dx$ would be spoken as "integral from x-ray equals alpha to bravo of foxtrot of x-ray plus golf of x-ray all delta x-ray end integral".

2. Parsing and Architecture of the TalkMaths system

In this section, we review previous work on parsing spoken mathematics using appropriate grammar and parser design, the majority of it having been carried out for the TalkMaths system.

This system is concerned with parsing suitably defined spoken forms of syntactically correct mathematical expressions, or else converting syntactically incorrect input to a format for display or further processing. Given a syntactically correct spoken mathematical expression, parsing will result in a parse tree that is useful for additional manipulations.

As mentioned by Fateman [18], it is difficult to specify a grammar that will parse the full set of mathematical terminology that one may typically encounter in research papers, text books or lecture notes. Hence for parsing our input, we will have to restrict ourselves to an appropriate sub-language, namely our standard of spoken mathematics, which has a well-defined vocabulary and relatively prescribed syntax, and which may be parsed with a context-free grammar, combined with some input pre-processing.

Fateman [19] adopts this approach in order to specify a subset of spoken mathematics, taking into account the fact that mathematical expressions use prefix, infix or suffix conventions and have occasionally ambiguous operator precedences.

In the first version of TalkMaths, described by Wigmore [16, 28], an attribute grammar [29, 30] for the Yapps2 parser generator is described that recognises spoken mathematics terminology at an elementary. However, it did not support incomplete input, which was a major drawback as users will tend to dictate more complex expressions in several distinct utterances with pauses.

As adding support for incomplete expressions to this attribute grammar appeared difficult, a new framework was developed by Attanayake [17, 27] that explored the use of special "mixfix" operators to construct our input language, and which used operator precedence parsing for the syntactical analysis of this language. To deal with ambiguities at lexical level, the XGLR parsing algorithm presented by Begel and Graham [31] was adapted for the operator precedence setting. Finally, robust error recovery strategies for the parser were designed, implemented and integrated in the subsequent version of TalkMaths, which by then had evolved into a web-based system [27].

The work presented in the next section is carried out in the context of the potential extension of TalkMaths to the mobile setting. Whereas Isaac *et al* [32] focus on an improved user interface design, this paper discusses an improved internal parsing algorithm that will be essential for use within the constraints of an environment such as a mobile app.

3. Incremental Parsing Algorithm

We now present our novel approach for incremental parsing of spoken mathematics. In order that spoken mathematics can be displayed or evaluated properly, operator precedence needs to be taken into account when parsing such content. The operator precedence parser developed by Attanayake [17, 27] does this, allows for incomplete

content, and handles errors in the user input. Although this is efficient for dictation from scratch, when the user wishes to edit an expression, the entire text needs to be reparsed before it may be rendered on screen or otherwise used, causing a perceptible delay for non-trivial expressions, which could cause problems for people using a web service and a mobile device. To ameliorate this, we need to be able to parse the input incrementally when the user performs an edit operation. Because we are working with mathematical expressions, our incremental parser needs to take operator precedence into account, producing a tree that represents the new expression that preserves operator precedence.

Incremental parsing was originally developed in the context of computer programming, to provide timely feedback to developers as they worked [33, 34]. Typically, the aim is to minimise the amount of source code that needs to be reparsed to reconstruct the parse tree, thus reducing the total time taken to produce an updated tree. While most incremental parsing approaches are designed to work with the types of grammar used by programming languages, some deal with operator precedence (OP) grammars, motivated either by the need to handle mathematical expressions within computer programs or specifically to parse mathematical content.

In the programming context, Lalonde & des Rivières [35] permit the parse tree for the OP grammar content to be built up without regard to OP, and then process it by rearranging nodes that violate the OP rules, until the tree is valid. Kaiser & Kant [36] approach the problem in the context of an editor for mathematical expressions, stipulating how the tree nodes are to be processed by the Lalonde & des Rivières transformations as a result of an editing action. The algorithms presented by Heeman [37] work on entire parse trees of the mathematical expressions, allowing any edit action to be achieved using two basic operations, *split* and *merge*.

All three approaches handle binary operators, brackets, and at least in concept, unary operators and functions, but only Heeman deals explicitly with incomplete expressions. However, none of them handle composite structures such as integrals, which are processed by the Attanayake [17] parser, effectively as *n*-ary operators. Our algorithm achieves this by extending those of Heeman [37] and making use of the Lalonde & des Rivières transformations [35]. It also improves upon Heeman's inefficient method of bracket matching, and has been implemented for use with the Attanayake [17] parser.

To illustrate the benefit of manipulating parse trees rather than reparsing a modified spoken expression, consider the situation where the user has already spoken "x-ray times echo to the power of x-ray" for xe^x and wants to add "plus charlie", giving $xe^x + c$. Without incremental parsing, as a first step, either the tree for xe^x would have to be converted back to words, or the spoken text version would have to be made available. Next, the new text needs to be concatenated to the original, and the result reparsed. Using incremental parsing, only the "plus charlie" needs to be parsed from scratch, and the tree for that concatenated with the original parse tree using the *merge* operation. For this particular example, that would consist of replacing the empty node to the left of "plus" with the tree for xe^x.

Work has started to evaluate the performance of these incremental parsing algorithms when compared to reparsing the entire text, and initial findings show that there is much less of a delay when carrying out edit operations than for reparsing. This is a crucial requirement for integration into a mobile app, and will significantly enhance the usability and hence the learning experience of the tool. In the future, we will thoroughly evaluate the full implementation of the algorithm in a mobile setting.

4. Conclusion

We have presented some of the potential benefits offered by speech input of mathematics, particularly for "mobile learning" and for disabled students. However, we also noted a number of difficulties associated with entering mathematical equations and formulae by speech, and proposed an approach that seeks to permit users to dictate mathematical expressions in a relatively natural way. The TalkMaths system was designed to implement this method for the dictation of mathematical equations and formulae; however, functionality is also required to allow users to edit and modify their mathematical content easily and efficiently. We have proposed a method for achieving this using an improved incremental parsing algorithm, based on that of Heeman [37], which is currently being evaluated for its performance parsing a set of typical equations.

In the future, we also hope to use these algorithms to permit creation and editing of computer programming code via spoken input. Both these systems – for dictation of mathematical expressions and of program code – should be of significant benefit disabled students, those wanting to engage in mobile learning, and "working on the go", using mobile devices which are less well suited to more conventional modes of input such as a keyboard and mouse.

References

[1] Deubel, P. Mobile Devices: Facing Challenges and Opportunities for Learning, 2009. THE Journal. https://thejournal.com/articles/2009/03/19/mobile-devices-facing-challenges-and-opportunities-for-learning.aspx (accessed May 26, 2016).

[2] Reat, L. P.; Carson, C. Mobile Devices Empower Students With Special Needs, 2014. Education World. http://www.educationworld.com/a_tech/apps-special-needs-disabilities-assistive-technology-students.shtml (accessed May 26, 2016).

[3] Kingston University London. TalkMaths. http://talkmaths.sourceforge.net/ (accessed May 26, 2016).

[4] Young, S. Large Vocabulary Continuous Speech Recognition: A Review. *IEEE Signal Processing Magazine* 1996, *13* (5), 45-57.

[5] Nuance Communications. Dragon NaturallySpeaking Professional. http://www.nuance.co.uk/for-business/by-product/dragon/dragon-for-the-pc/dragon-professional/ (accessed May 31, 2016).

[6] Carnegie Mellon University. Sphinx-4 A speech recognizer written entirely in the Java(TM) programming language. http://cmusphinx.sourceforge.net/sphinx4/ (accessed May 31, 2016).

[7] Google. Google Speech Demo. http://www.google.com/intl/en/chrome/demos/speech.html (accessed May 31, 2016).

[8] Holmes, J.; Holmes, W. *Speech Synthesis and Recognition*, Second Edition; Taylor and Francis, 2000.

[9] Metroplex Voice Computing, Inc. mathtalk.com. http://www.mathtalk.com/ (accessed May 31, 2016).

[10] Elliott, C.; Bilmes, J. A. Computer Based Mathematics using Continuous Speech Recognition. *CHI 2007 Workshop on Striking a C[h]ord: Vocal Interaction in Assistive Technologies, Games and More*, San Jose, CA, USA, 2007.

[11] Bernareggi, C.; Brigatti, V. Writing Mathematics by Speech: A Case Study for Visually Impaired. *Proceedings of ICCHP 2008 (11th International Conference on Computers Helping People with Special Needs)*, 2008; pp 879-882, Springer.

[12] Hanakovič, T.; Nagy, M. Speech Recognition Helps Visually Impaired People Writing Mathematical Formulas. *Computers Helping People with Special Needs*, 2006; pp 1231-1234.

[13] Attanayake, D. R.; Hunter, G.; Denholm-Price, J.; Pfluegel, E. Novel multi-modal tools to enhance disabled and distance learners' experience of mathematics. *International Journal on Advances in ICT for Emerging Regions (ICTer)* 2013, *6* (1), 26-36.

[14] Chang, L. A. *Handbook for spoken mathematics (Larry's speakeasy);* Lawrence Livermore National Laboratory, University of California, USA, 1983.

[15] Raman, T. V. *Audio system for technical readings* (ASTeR); Springer Verlag: Berlin, 1998.

[16] Wigmore, A. M. *Speech-Based Creation and Editing of Mathematical Content;* PhD Thesis; Kingston University, UK: Kingston-upon-Thames, UK, 2011.

[17] Attanayake, D. *Statistical Language Modelling and Novel Parsing Techniques for Enhanced Creation and Editing of Mathematical E-Content Using Spoken Input;* PhD Thesis; Kingston University: London, UK, 2014.

[18] Fateman, R. *2-D Display of Incomplete Mathematical Expressions*, 2006. http://www.eecs.berkeley.edu/~fateman/papers/dispbad.pdf (accessed May 31, 2016).
[19] Fateman, R. *How can we speak math* ?, 2009 http://www.eecs.berkeley.edu/~fateman/papers/speakmath.pdf (accessed May 31, 2016).
[20] Guy, C.; Jurka, M.; Stanek, S.; Fateman, R. *Math Speak & Write, a Computer Program to Read and Hear Mathematical Input;* Electrical Engineering and Computer Sciences Department, University of California, Berkeley, USA, 2004.
[21] Wigmore, A. M.; Hunter, G. J. A.; Pfluegel, E.; Denholm-Price, J. TalkMaths: A speech user interface for dictating mathematical expressions into electronic documents. *2nd ISCA Workshop of Speech and Language Technology in Education (SLaTE 2009)*, 2009; P 3-4.
[22] Burnard, L. *User's Reference Guide for the British National Corpus*, 1995. http://www.natcorp.ox.ac.uk/ (accessed May 31, 2016).
[23] University of Southampton, UK, *STEMReader*. https://stemreader.org.uk/ (accessed May 24, 2016).
[24] Schleppenbach, D. A. 2004 *MathSpeak Initiative*. http://www.gh-mathspeak.com/ (accessed May 31, 2016).
[25] Kacorri, H.; Riga, P.; Kouroupetroglou, G. EAR-Math: Evaluation of Audio Rendered Mathematics. In *Universal Access in Human-Computer Interaction. Universal Access to Information and Knowledge: 8th International Conference, UAHCI 2014*, 2014; pp 111-120, held as part of *HCI International 2014*, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part II, Springer International Publishing
[26] Wolfram Mathematica. SpokenString - *Wolfram Mathematica 9 Documentation*, 2014. SpokenString - Wolfram Mathematica 9 Documentation. http://reference.wolfram.com/mathematica/ref/SpokenString.html (accessed May 24, 2016).
[27] Attanayake, D. R.; Denholm-Price, J.; Hunter, G.; Pfluegel, E. TalkMaths over the web: a web-based speech interface to assist disabled people with mathematics. *Proceedings of the Institute of Acoustics*, 2014; *36 (3)* pp 443-450.
[28] Wigmore, A.; Hunter, G.; Pfluegel, E.; Denholm-Price, J.; Binelli, V. Using Automatic Speech Recognition to Dictate Mathematical Expressions: The Development of the 'TalkMaths' Application at Kingston University. *Journal of Computers in Mathematics and Science Teaching* 2009, *28* (2), 177-189.
[29] Knuth, D. E. The genesis of attribute grammars. In *Attribute Grammars and Their Applications;* Springer, 1990; pp 1-12.
[30] Kastens, U. Ordered attributed grammars. *Acta Informatica* 1980, *13* (3), pp 229-256.
[31] Begel, A.; Graham, S. L. XGLR - an algorithm for ambiguity in programming languages. *Science of Computer Programming* 2006, *61* (3), pp 211-227.
[32] Isaac, M.; Pfluegel, E.; Hunter, G.; Denholm-Price, J. Intuitive NUIs for speech editing of structured content (work in progress). *26th Annual Workshop of Psychology of Programming Interest Group, PPIG 2015*, Bournemouth, UK, 2015; pp 1-5.
[33] Ghezzi, C.; Mandrioli, D. Incremental Parsing. *ACM Transactions on Programming Languages and Systems* 1979, *1*, pp 58-70.
[34] Ghezzi, C.; Mandrioli, D. Augmenting Parsers to Support Incrementality. *Journal of the ACM* 1980, *27* (3), pp 564-579.
[35] Lalonde, W. R.; des Rivieres, J. Handling Operator Precedence in Arithmetic Expressions with Tree. *ACM Transactions on Programming Languages and Systems* 1981, *3*, pp 83-103.
[36] Kaiser, G. E.; Kant, E. Incremental parsing without a parser. *Journal of Systems and Software* 1985, pp 121-144.
[37] Heeman, F. C. Incremental parsing of expressions. *Journal of Systems and Software* 1990, pp 55-69.