




Rule based intelligent system verbalizing mathematical notation

Agnieszka Bier¹  · Zdzisław Sroczyński¹

Received: 1 October 2018 / Revised: 15 April 2019 / Accepted: 10 June 2019 /
Published online: 4 July 2019
© The Author(s) 2019

Abstract

An adaptive and adaptable multi-purpose math-to-speech translation system is proposed in the paper. Along with the detailed presentation and design approach for the core math-to-speech translation system, exemplary output and tests are discussed. A scripting extension, providing flexibility of the system and enabling the user to adjust the output translations to his/hers preferences is incorporated into the presented solution. Some unit-tests and adaptable versions of translation rule sets are elaborated and evaluated by means of standard measures of machine translation quality. Designed as a flexible self contained solution, the presented system may be applied for various purposes such as e-learning audio presentation systems, educational tools for non-native speakers or visually impaired people, and may be easily adjusted to different national languages.

Keywords Text-to-speech interface · Verbalizing mathematical notation · Adaptive interface · Visually impaired learning assistance

1 Introduction

1.1 Motivation

A proper and effective Human-Computer Interaction (HCI) model is an essential and challenging issue in the design of modern computer based systems, especially those intended for e-learning [21, 23], mobile devices [45] or support for people with special needs [26, 33, 35, 42], including elderly ones [34]. The research in this area takes into account many different users' needs, and main conclusions point out the need to increase flexibility and adaptivity of user experience [6].

These challenges are especially important in the field of computer interaction with visually impaired people [14], who are unable to use the common graphical user interface and

✉ Agnieszka Bier
agnieszka.bier@polsl.pl

Zdzisław Sroczyński
zdzislaw.sroczynski@polsl.pl

¹ Faculty of Applied Mathematics, Institute of Mathematics, Gliwice, 44-100, Poland

require specialized assistance [9, 12], such as adaptable screen content presentation, screen readers [31], touchscreen systems [7] or specialized Braille languages interfaces [15, 16]. One of the key difficulties in this area is establishing technique for proper input and output of mathematical content, such as formulae or equations [25, 27]. The nature of the mathematical notation obstructs semantic reception of the equations and hence often becomes a significant barrier for visually impaired people, who then need the assistance of peer readers or an automated speech system. Unfortunately, most of the present alternative input methods for mathematics do not provide versatile help in editing complex equations.

1.2 Problem of verbalizing mathematics and available solutions

Mathematical notation is based on a variety of symbols, graphs and diagrams, usually describing abstract objects that are impossible to present linearly without extensive usage of parenthesis. This non-linearity becomes the main challenge in designing an accurate tool for automatic translation of mathematical notation. Moreover, the verbalization of math notation requires keeping the accuracy while the brevity of the message increases in order to make it still understandable for the audience. Another important issue in machine translation of mathematics is the rigidity of translation rules that are typically used regardless the context, language or particular subject. The issues which concern grammar and templates for processing of the mathematical notation are elaborated in [43] and [24].

Different approaches have been used for automatic conversion to/from spoken form of mathematical notation. One possible solution for the system to read (or write) mathematics in an understandable and proper way is to equip it with a tool for machine translation from mathematical symbolic notation to spoken language [13] and the speech synthesis [18] which often needs additional efforts for non-English national languages [36]. Most of the existing TTS systems intended for mathematics are focused on particular type of input (like \LaTeX or MathML) and use a set of pre-defined translation rules in the translation process. Such rigid approach may lead to superfluous translations, which are not natural and comprehensible for the audience. Exemplary systems of this type include AudioMath [20], a tool applying rigid verbalization rules stored in databases, Math Speak & Write [40], a solution integrating Lisp with MS Windows Speech API, some assistive solutions for the visually impaired, like TalkMaths [3, 48], a system for dictating mathematical contents or MathSpeak [30], a tool for \LaTeX automatic translation.

Other approaches include for example math OCR for recognition of graphically presented expressions followed by an analysis of their 2D structure to linearize the presentation for further processing [29], or a \LaTeX package for converting \LaTeX commands into their natural language counterparts [1].

It should be emphasized that majority of the existing solutions support only English language, including the commercial tools like MathPlayer plugin for MS Office or MathTalk. Due to grammar and semantics these solutions cannot be directly used for other national languages. There are dedicated tools proposed for some national languages, like Slovak [10], Portuguese [20], Chinese [46] or Thai [50]. Moreover, most of the presented systems are developed and tested on relatively simple examples and do not support more complex expressions (see also [2, 4, 5]). Some of them require manual pre-editing of the alternative content by the specialized operator [1, 28]. There is a lack of a flexible solution, which would enable switching the languages of translation, setting the mathematical context and editing translation rules.

In the view of the above introduction, the aim of the work presented within this paper was to develop a math-to-speech system, which allows for adapting the translation rules to

individual user's needs. The proposed solution extends the basic math-to-speech system [8], which was originally developed for translation of mathematical expressions into Polish and was based on hard-coded set of translation rules. Multiple tests run on the system revealed the need to develop a solution, which would offer flexibility in rule setting. The system has been re-designed and extended by the universal scripting language, providing a unique possibility to edit, correct or alter the machine translation algorithm by the final user. The implementation details and the unit-testing results of the proposed system are presented in the subsequent sections. In addition, the obtained values of standard measures for translation quality evaluation are provided as a reference for the performance quality of the system. Also, the conclusions from questionnaires filled by the users performing typical tasks with the use of introduced scripting tools are discussed.

2 Material and methods

2.1 The core math-to-speech system

The basic math-to-speech system described in [8] performs machine translation with the use of hard-coded rules. It verbalizes mathematical expressions defined in various formats: web math notation (MathML), \LaTeX and EQED (introduced in the included visual equation editor) to linear textual form of natural spoken language in Polish. As Text-To-Speech (TTS) engines are built-in into common computer operating systems or available at low cost, the main problem is the proper linear composition of the meaning of mathematical expressions. The resulting translation should be suitable for speaking and understandable without ambiguity, and should take into account the personal preferences of the user (e.g. the permissible level of simplification, dialect or context).

In the basic math-to-speech system the following types of inputs have been implemented both in the standard/full (SM) and the optimized/simplified (OM) mode: quotients, powers, roots, brackets, indices, integrals, sums, products and limits. The two modes SM and OM were introduced in order to adjust the produced translation to particular user's requirements. In SM the system produces the full linear translation based on the set of strict translation rules, which sometimes may sound superfluous especially in the case of relatively simple and short expressions. This excess is reduced in the OM, which implements a number of simplified contextual translation rules at the expense of translation explicitness. Detailed presentation of the system performance in Polish interface is given in [8]. The system integrates a math notation visualizing engine, interactive equation editor desktop application and a set of basic hard-coded translation rules. It allowed to distinguish and fine-tune the rule sets for different areas of mathematics, however the process was time consuming due to the complexity of testing procedure and the need of subsequent recompilation. The rule sets were determined for Polish language only.

2.2 Scripting capabilities in the math-to-speech translation

Hard-coding of the basic math-to-speech system made it easier to implement and interact with internal data structures containing mathematical notation. However, every modification of the translation system involved the need for recompile the application. This way the math-to-speech interface was adaptable only within the borders of implemented rules.

Introducing scripting language for the translation process opens up new opportunities, such as dynamic exchange of rules, additions of novel dialects and national versions. After

the re-implementation of the existing rules we were able to add some valuable factors to the math-to-speech engine. First, the English version of the translation was added. Exemplary translations obtained for the formula

$$\sum_{i=1}^{n^2} \left(\frac{1}{2} + \sqrt[n]{\sqrt{i}} \right)^3$$

in both modes of English version (standard SM and optimized OM) are presented below:

SM: *summation from i equals 1 to n squared of open parenthesis fraction: numerator: 1, denominator: 2, end of fraction plus root of degree n of square root of i, end of root, , end of root, closed parenthesis cubed end of summation*

OM: *summation from i equals 1 to n squared of open parenthesis 1 half plus nth root of square root of i, end of root, , end of root, closed parenthesis cubed end of summation*

Secondly, we added a universal set of test equations with their translations and provided fast and convenient way of test automation for the math-to-speech process. In the end we provided the possibility to extend the translation engine with new, user-defined rule sets, which interact with GUI of equation editor and can be chosen by the user on-the-fly.

This way we have extended the adaptivity and adaptation functions of the math-to-speech system. These extensions were tested by several users – mathematicians without special knowledge in script programming. The results of the surveys seem promising, as all the respondents were satisfied with the efficiency of proposed solution. In this case the HCI is both complex enough to provide wide and comprehensive solutions, and easy to learn and implement on the other hand.

3 Theory and implementation

There are many popular scripting languages, as for example Visual Basic, PHP, PascalScript, Javascript, Python and Lua. The chosen solution for our project was Lua: easy to learn with notation close to Pascal, with small footprint, well designed to interact with other languages and having enough ready libraries to develop even complex applications. Lua language was indeed developed with software automation in mind. It is also widely used in production environments, which ensures stability and maturity of the project. The best known implementations of Lua are gaming platforms: Corona SDK, Moai SDK or LÖVE SDK. Lua is also widely used in science projects, especially as a tool for definition of complex, rule-based systems designed to process natural language [38] and for general software automation, for example in LuaTeX project [41].

Inside the core of the equation editor developed by our team and used in all experiments (“Equation wizard”), the mathematical notation is stored as a tree of objects. Therefore, the translation process is a top-down traversal of this internal data structure, verbalizing the expression recursively. We decided to leave the traversal in the main application and call specialized Lua functions to translate particular elements of mathematical notations. This way we did not need to rewrite big parts of the engine and could keep the Lua functions short and easy to learn or modify. Moreover, the multiplatform software development tools used for “Equation wizard” project (i.e. RAD Studio from Embarcadero), give the opportunity to port the core engine and create user experience for popular mobile operating systems without significant effort [44]. The scheme in the Fig. 1 shows these two levels of the

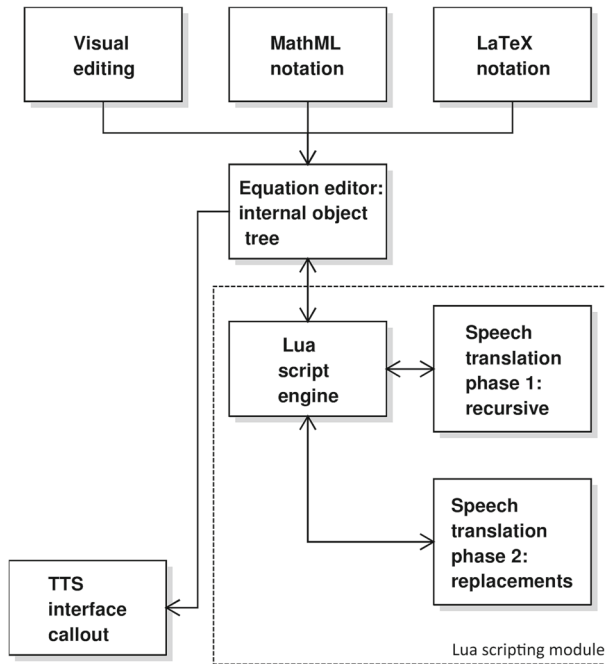


Fig. 1 The flowchart for the translation of the mathematical notation into speech. Two translation phases: recursive (the traversal of the internal object tree, and replacements (dictionary conversion) performed by scripting engine (dashed box)

translation process. In the second phase a set of additional subsequent text replacements is performed, allowing simple substitution of names for Greek letters and other basic symbols.

The translation algorithm operates recursively whenever a nested part is found in the translated expression. Nevertheless, the overall execution time remains linear with respect to the total number of atomic symbols to be translated. Some additional processing is required only when the parser has to back propagate to the previous symbol, however this results in only one additional operation to be performed and has no remarkable influence on the overall complexity of the translation algorithm. The system is designed to be an efficient interactive tool – the processing time of a relatively long formula is of no significance in terms of user experience as the operation is real-time for a typical PC with Windows 10. However, designed for interaction, the solution in current form is rather not scalable to process big data sets – for these kind of applications hard-coded rules ensure better efficiency of the translation algorithms.

3.1 Details of the Lua scripting implementation

The syntax of Lua language is similar to Pascal with some exceptions: 1. there are no typed variables, 2. every code block is determined by `end` keyword (there is no `begin` keyword). Lua deals with long unicode character strings very well, so it is suitable for our goals.

Several Lua functions with fixed names were introduced for each of the translation rule sets. Every translation rule set is stored in a separate external text file, easy to edit. The rule set defined in a single Lua file is intended to be used for translation in the particular

national language, mathematical dialect or verbalization approach (like standard/optimized modes).

User can define additional help functions, which can be used to solve particular translation problems. Fix-named functions share the following signature:

```
Tename_of_element(indH, indL, Body, Prev)
```

where `name_of_element` can be for example `root`, `int`, `brackets` etc. The translated descriptions of high and low index, main part of the sub-expression (the body), and previous element should be passed into respective parameters.

In every case, `name_of_element` is an equivalent of the internal object in the "Equation wizard" editor core, representing particular mathematical symbol with its context. It means, that for example `power` should have sub-expression in superscript (exponent), but `integral` should have upper and lower bounds. The elements which are not obligatory can be passed to translation functions as empty strings.

Therefore, the `indH` and `indL` parameters have the context dependant meaning, for example `indH` is used to describe numerator in fractions, root degree, exponent or upper bound of the definite integral. Similary, `indL` is used to denote lower index, denominator in fractions or lower bound of the definite integral (sum, product, limit etc.)

Providing verbalized value of the previous element (`prev` parameter) allows to assure the contextual interpretation of the appropriate parts of the expression. For example it is used in the `TEfrac` function to determine if the conjunction "and" should be placed before the actual verbalization of the fraction (see Listing 1). This way we receive the correct "two and fraction..." for $2\frac{a}{b}$.

Common signature of translation functions simplifies the interface between the core tree-like data structure used by "Equation wizard" editor and Lua interpreter. It is realized through the single internal function:

```
function LuaInvoke(L: lua.State; fname, indH, indL, Body,
    Prev: ← String): String;
```

which calls Lua function named `fname` with the following parameters. Parameter `fname` should certainly correspond with the name of the function to call from Lua source file (rule set), i.e. `Tename_of_element`. The two next parameters and their contextual usage was discussed above.

The most seldom used parameter is `Prev`, so the extra wrapper function fills it with default empty string:

```
function LuaSpeak(fname, indH, indL, Body: String; Prev :
    String ← = ''):String;
```

Wrapper function `LuaSpeak` initializes Lua interpreter, passes Lua source code (rule set) with translation definitions according to chosen context and national language and calls `LuaInvoke` in the end. As a result we receive the translation actually performed with the

Listing 1 "Prev" parameter usage in the beginning of the Lua translation function for fractions

```
function TEfrac(indH, indL, Body, Prev)
    result = ''
    num = trim(indH);
    den = trim(indL)
    if isInt(trim(Prev)) then
        result = ' and '
    end
    ...
```

use of Lua functions (describing translation rules), defined in the external files. As these files are simple text-files with source code in Lua, they are easy to edit, change, correct and extend, without the need to recompile the whole editor application. The GUI of "Equation wizard" editor scans for the translation rule set files automatically, so there is no need for re-run of the editor after editing the Lua part. This way our solution offers the convenient way to add national languages, dialects or user-preferred versions.

The mutual relations between the modules of the solution developed during our experiments is shown in Fig. 1. The main translation loop traverses recursively the internal tree-like data structure stored in the "Equation wizard" editor core calling EqSpeak function. Math-to-speech translation of every particular symbol or subexpression requires a call to the appropriate Lua function. This translation procedure can be described alternatively by the simplified pseudocode from Algorithm 1.

Algorithm 1 The pseudocode for the math-to-speech translation function EqSpeak (contextual issues and Prev parameter details intentionally omitted for simplicity).

```

function EqSpeak(parameter: equation element (ee) ) -> verbalization string
  result = ""
  if ee is Plain_symbol or Number then
    result = ee
  else if ee is Fraction then
    result = LuaSpeak('TEfrac', EqSpeak (numerator), EqSpeak(denominator), "")
  else if ee is Root then
    result = LuaSpeak('TEroot', EqSpeak(degree), "", EqSpeak(radical))
  else if ee is Integral then
    result = LuaSpeak('TEint', EqSpeak(upper bound), EqSpeak(lower bound), EqSpeak(body expression))

  else if ...
  else if ee is List then
    for all subsequent equation elements do
      result = result + EqSpeak(element)
    end for
  end if return result

```

Additionally, every file with source code for the translation rule set may contain special functions providing automatic tests of any number of equations. This way we help to build stable solutions and avoid regression errors while editing functions with definitions of translation rules.

3.2 Examples of the math-to-speech translation Lua functions

The Listing 2 presents an example of the translation function for roots. This particular example is designed for Polish optimized version i.e. version taking into account context and the level of complication. The implementation differentiates between square roots and roots of given degree.

English translation for the same element takes into account roots of degree 2, 3 and the other, as well as empty degree (which traditionally refers to square root – see Listing 3). These two very simple examples illustrate the elastic approach for the translation resulting in possibilities to build adaptive versions of the math-to-speech system.

```

function TErroot(indH, indL, Body, Prev)
  if trim(indH)==' ' then
    wyn = ' pierwiastek z ' .. Body .. ', koniec pierwiastka, '
  else
    wyn = ' pierwiastek stopnia: ' .. indH .. ' pod ←
    pierwiastkiem: ' .. Body .. ', koniec pierwiastka, '
  end
  return wyn
end

```

Listing 2 Lua translation function for root symbols (Polish optimized version)

Much more complex Lua functions describe for example translations for superscripts or brackets. They deal with length of the sub-expressions and check if parts of equations are integer numbers or operators.

3.3 User interface for the translation module in the host application

User experience for the functionality described above is provided through the interactive dialog window, which contains a list of available translations with their descriptions. The short names used in listbox, as well as longer informative commentaries are loaded dynamically from the Lua source with the use of predefined functions TEGetName and TEGetDescription (Fig. 2).

Keeping many versions of translation rule sets can be very helpful, especially to provide exact compatibility with the national or regional regulations and de-facto standards. As one of the most obvious usages of the math-to-speech interfaces is the extended support for visually impaired people, it is worth to note how easy it becomes to develop rules for phonetic alphabet speak-out of one letter variables (i.e. "alpha" for "a", "bravo" for "b", "charlie" for "c", etc. [19])

3.4 Automated tests of the math-to-speech algorithms

Software automation with the use of Lua scripting language helped also to develop a set of tests. Manually triggered by the user, they check every translation function against pattern equations and their reference spoken versions. The tests do not take into account whitespaces, which do not matter for verbal effects in the TTS engines. This kind of testing helps to avoid simple regression errors while the translation engine rule sets are edited, so it is an equivalent of unit-tests in common programming languages.

```

function TErroot(indH, indL, Body, Prev)
  if (trim(indH)==' ') or (trim(indH)=='2') then
    wyn = ' square root of ' .. Body .. ', end of root, '
  else
    if (trim(indH)=='3') then
      wyn = ' cube root of ' .. Body .. ', end of root, '
    else
      wyn = ' .. trim(indH) .. 'th root of ' .. Body .. ', ←
      end of root, '
    end
  end
  return wyn
end

```

Listing 3 Lua translation function for root symbols (English optimized version)

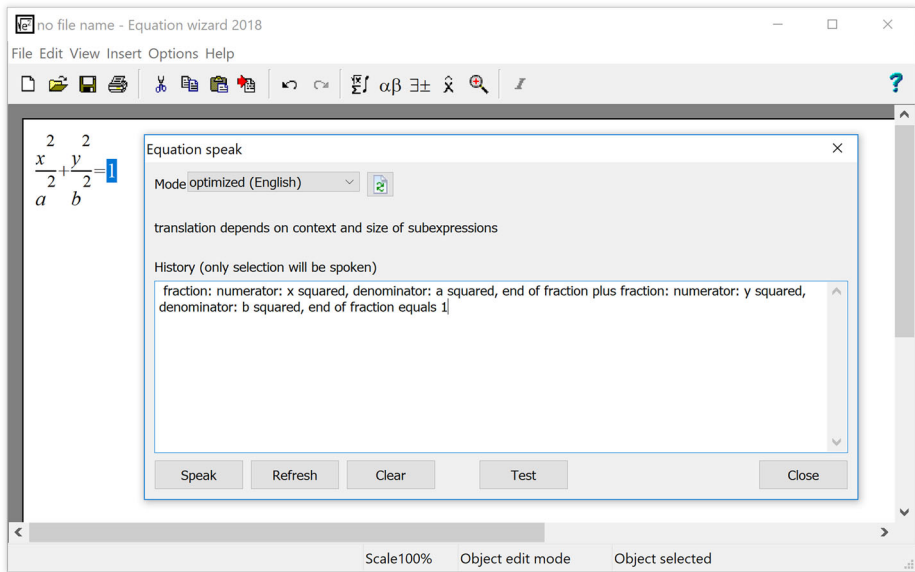


Fig. 2 Graphical user interface for the translation module. The main "Equation wizard" window in the background

The corpus of the test functions checks all the elements and subexpression types, which can appear in the equations understandable by the developed system. There are many simple and relatively short examples in the set, as well as a few larger examples, checking nested parts of equations.

Some exemplary English optimized translations for roots are presented in the Table 1. Every test equation is assigned to its description in the internal "Equation wizard" (EQED) notation and reference translation.

Table 1 Exemplary mathematical expressions used for automatic translation testing

Symbolic notation	EQED notation	Spoken translation reference version (English)
\sqrt{a}	<code>\EQEDroot {\EQEDplain{}} \EQEDplain{a}}</code>	<i>square root of a, end of root,</i>
$\sqrt[2]{a}$	<code>\EQEDroot {\EQEDnplain {2}}{\EQEDplain{a}} }</code>	<i>square root of a, end of root,</i>
$\sqrt[3]{a}$	<code>\EQEDroot {\EQEDnplain {3}}{\EQEDplain{a}} }</code>	<i>cube root of a, end of root,</i>
$\sqrt[5]{a}$	<code>\EQEDroot {\EQEDnplain {5}}{\EQEDplain{a}} }</code>	<i>5th root of a, end of root,</i>

The interaction between main application GUI and Lua test code is provided by the special call-back function `TEtest`. This function responds to the following commands:

1. `GETREF` with the index of test in the parameter: returns EQED notation for the test equation,
2. `GETSPEECH` with the index of test in the parameter: returns reference verbal translation according to the chosen translation rule set (i.e. given Lua file),
3. `CHECK` with the index of test in the first parameter, verbal translation to compare in the second parameter and possibly a flag `NOPUNCTATION` in the third parameter: returns boolean value for passed or failed test.

The source data for tests is stored in two-dimensional Lua table. Thanks to Lua special syntax, it is possible to embed raw multi-line text data into the source code easily. So, the EQED descriptions can be directly copied from editor to the test suite.

The test corpus for optimized English includes several dozens of equations. The simplest samples in the set contain just a few symbols. They are used for testing particular rules in the translation system. On the other hand, the most complex expressions contain over 50 mathematical symbols and they are suitable to test nested subexpressions. The sample output log from this test suite is shown by the Listing 4.

4 Results and discussion

4.1 System transcriptions vs human perception

In order to verify the established translation rules, a spoken outcome of few sample formulae has been compared with human perception abilities. A group of 41 engineering students has been asked to write down the original mathematical formulae, basing on the spoken transcriptions produced by our system. The students took the tests in smaller groups of 6, 8 and 27 and they heard the translation only once. There were seven tasks (formulae) to be performed in the test (instead of transcriptions we give only the symbolic formulae):

(a) $\sqrt[n]{\frac{a^4}{2}}$,

```

...
SOURCE=\EQEDroot{\EQEDplain{}}{\EQEDplain{x}}
TRANSLATION= square root of x , end of root ,
REFERENCE=square root of x , end of root ,
***** test No 31 ok *****
SOURCE=\EQEDroot{\EQEDnplain{2}}{\EQEDplain{x}}
TRANSLATION= square root of x , end of root ,
REFERENCE=square root of x , end of root ,
***** test No 32 ok *****
SOURCE=\EQEDroot{\EQEDnplain{3}}{\EQEDplain{x}}
TRANSLATION= cube root of x , end of root ,
REFERENCE=cube root of x , end of root ,
***** test No 33 ok *****
SOURCE=\EQEDroot{\EQEDnplain{5}}{\EQEDplain{x}}
TRANSLATION= 5th root of x , end of root ,
REFERENCE=5th root of x , end of root ,
***** test No 34 ok *****
***** all tests: 34 **** tests failed: 0 *****

```

Listing 4 A sample from automatic test log of translation from structural math notation to spoken English

- (b) $\left(\frac{2}{x}\right)^5$,
 (c) $\int f(x+5)dx$,
 (d) $\sqrt{\frac{2}{3}} + 1$,
 (e) $a_{2n+1}^3 = b_{2n}^{3n-4}$,
 (f) $(a+b+c)^2 - (a-b-c)^2 = 4(ab+ac)$,
 (g) $\left[\frac{\sqrt[4]{3x^2-5}}{(x-y)^{x+y}}\right]$.

All answers were then collected and analyzed in terms of possible dissimilarities with the symbolic formulae. A majority (nearly 83%) of the answers were absolutely correct, i.e. the formulae written by respondents were exactly those entered to the math-to-speech system. The worst results were observed for the formula (e), which was successfully recovered only by a half of students. The mistakes were usually made with lower and upper indices, which refer strongly to the visual form of the formula and therefore are hard to process linearly by human brain. Also, in a total number of 15 answers the mistakes related to bracket placing was observed. On the other hand, formula (f) was correctly recovered by all students, and the general success rate of 82, 9% can be considered satisfactory. Table 2 presents the detailed results for all seven formulae.

The survey has proved the usability of the proposed system for educational purposes. Of course, the perception of people with visual impairment may be different, hence an additional test of our system for these people is planned.

4.2 Word metrics for translation quality evaluation

In order to evaluate the quality of the output from the math-to-speech engine, one should decide on metrics for measuring the distance from the model transcript of the given expression to the respective output produced by the system. In our study we used two standard metrics independently: WER (Word Error Rate) and BLEU (BiLingual Evaluation Understudy). WER is the metric based on the implementation of Levenshtein distance at the words level. It is defined as:

$$WER = \frac{S + D + I}{N}$$

where S – the number of substitutions, D – the number of deletions, I – the number of insertions and N – the number of words in the reference. By definition, smaller WER indicates better similarity (the reference and candidate strings are closer).

Table 2 Accuracy of recovery of the original formulae by students

Formula	Error-free transcriptions	Transcriptions with errors	Accuracy rate
a	31	10	75.61%
b	40	1	97.56%
c	36	5	87.80%
d	35	6	85.37%
e	20	21	48.78%
f	41	0	100.00%
g	35	6	85.37%
Total	238	49	82.93%

BLEU is the contemporary standard and widely used metric for evaluation of machine translation systems [49] that allows to consider more than one reference translation. Although BLEU is mostly used for analysis of statistical translation systems, it can be also implemented in assessment of rule-based translations [17]. In the included examples, the standard BLEU values are given. The are calculated as

$$BLEU = BP \cdot \sqrt[4]{\prod_{n=1}^4 P_n}$$

where P_n denotes the precision of n -grams (tuples of n words), defined as

$$P_n = \min(\max(\# \text{ n-grams in reference file}), \# \text{ n-grams in candidate file}),$$

BP – Brevity Penalty, defined as

$$BP = \begin{cases} 1 & \text{for } c > r \\ e^{1-\frac{r}{c}} & \text{for } c \leq r \end{cases},$$

c – number of words in candidate string, r – the number of words in reference string. Standard BLEU is a number between 0 and 1 and the higher BLEU value is, the more the compared strings are similar. For details of the BLEU metric calculations we refer the reader to [32].

4.3 Quality evaluation for different translation modes

A few examples demonstrating the effectiveness of equation reader run with two versions of modified Lua scripts: Standard English (SM) and Optimized English (OM) are listed below. The respective WER and BLEU metrics for the quality of translation performed in each case in comparison to the assumed natural language transcription(s) of the exemplary formulae (referred to as Spoken English) were calculated in each case.

The assumed verbalization are based on own elaboration and literature sources, including fundamental book [11], as well as some academic reports [19, 22, 37] and textbooks [47].

The following examples present the capabilities of the verbalization algorithm and may be used as reference for comparison with other solutions. The results are classified into basic categories of mathematical expressions implemented in our system: fractions, powers, roots, summation/products, subscripts and special symbols. In the end some miscellaneous formulae containing expressions of various categories are given.

1. Fractions:¹

$$\frac{n-1}{n+1} + 1\frac{1}{2} + 5\frac{23}{3} - \frac{8}{4} - \frac{2}{1} + \frac{71}{5} + \frac{2n}{3m}$$

- Reference: *fraction: numerator: n minus 1, denominator: n plus 1, end of fraction plus 1 and a half plus 5 and 23 thirds minus 8 quarters minus 2 over 1 plus 71 over 5 plus 2n over 3m*
- OM: *fraction: numerator: n minus 1, denominator: n plus 1, end of fraction plus 1 and 1 half plus 5 and 23 thirds minus 8 quarters minus 2 over 1 plus 71 over 5 plus 2n over 3m; BLEU = 0.937, WER = 0.0238*

¹all visualized exemplary test equations intentionally placed in black boxes

- SM: fraction: numerator: n minus 1, denominator: n plus 1, end of fraction plus 1 and fraction: numerator: 1, denominator: 2, end of fraction plus 5 and fraction: numerator: 23, denominator: 3, end of fraction minus fraction: numerator: 8, denominator: 4, end of fraction minus fraction: numerator: 2, denominator: 1, end of fraction plus fraction: numerator: 71, denominator: 5, end of fraction plus fraction: numerator: $2n$, denominator: $3m$, end of fraction; BLEU = 0.2133, WER = 1.6667

2. Powers:

$$2^i + 2^{i+1} + 2^k + 2^{3^2}$$

- Reference: 2 to the power of: i , plus 2 to the power of: i plus 1 end of exponent, plus 2 to the n th plus 2 to the power of: 3 squared
- OM: 2 to the power of: i end of exponent, plus 2 to the power of: i plus 1 end of exponent, plus 2 to the n th plus 2 to the power of: 3 squared end of exponent; BLEU = 0.8132, WER = 0.1667
- SM: 2 to the power of exponent i end of exponent plus 2 to the power of exponent i plus 1 end of exponent plus 2 to the power of exponent n end of exponent plus 2 to the power of exponent 3 squared end of exponent; BLEU = 0.4354, WER = 0.4722

$$\sum_{i=-3}^9 x^i = x^{-3} + x^{-2} + x^{-1} + x^0 + x^1 + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9$$

- Reference: summation from i equals minus 3 to 9 of x to the power of i equals x to the power of minus third, plus x to the power of minus second, plus x to the power of minus first, plus x to the zeroth, plus x to the first, plus x squared plus x cubed plus x to the fourth, plus x to the fifth, plus x to the sixth, plus x to the seventh, plus x to the eighth, plus x to the ninth
- OM: summation from i equals minus 3 to 9 of x to the power of: i end of exponent, end of summation equals x to the power of minus third, plus x to the power of minus second, plus x to the power of minus first, plus x to the zeroth, plus x to the first, plus x squared plus x cubed plus x to the fourth, plus x to the fifth, plus x to the sixth, plus x to the seventh, plus x to the eighth, plus x to the ninth; BLEU = 0.8987, WER = 0.0842;
- SM: summation from i equals minus 3 to 9 of x to the power of exponent i end of exponent end of summation equals x to the power of exponent minus 3 end of exponent plus x to the power of exponent minus 2 end of exponent plus x to the power of exponent minus 1 end of exponent plus x to the power of exponent 0 end of exponent plus x to the power of exponent 1 end of exponent plus x squared plus x cubed plus x to the power of exponent 4 end of exponent plus x to the power of exponent 5 end of exponent plus x to the power of exponent 6 end of exponent plus x to the power of exponent 7 end of exponent plus x to the power of exponent 8 end of exponent plus x to the power of exponent 9 end of exponent; BLEU = 0.3324, WER = 0.8316;

3. Roots:

$$\sqrt{x} + \sqrt[3]{x-1} + \sqrt[4]{\frac{2}{5}} + \sqrt[9]{2} + \sqrt[123]{7x+1} + \sqrt[n+1]{n-1}$$

- Spoken 1: square root of x plus cube root of x minus 1, plus 4th root of 2 over 5 plus 9th root of 2 plus root of degree 123 of 7x plus 1, plus root of degree n plus 1 of n minus 1
- Spoken 2: square root of x plus cube root of the quantity: x minus 1, plus 4th root of 2 over 5 plus 9th root of 2 plus root of degree 123 of the quantity: 7x plus 1, plus root of degree n plus 1 of the quantity n minus 1
- OM: square root of x , end of root, plus cube root of x minus 1 , end of root, plus 4th root of 2 over 5 , end of root, plus 9th root of 2 , end of root, plus root of degree 123 of 7x plus 1 , end of root, plus root of degree n plus 1 of n minus 1 , end of root, ; BLEU = 0.5248, WER1 = 0.6; WER2 = 0.6604;
- SM: square root of x , end of root, plus cube root of x minus 1 , end of root, plus root of degree 4 of fraction: numerator: 2, denominator: 5, end of fraction , end of root, plus root of degree 9 of 2 , end of root, plus root of degree 123 of 7x plus 1 , end of root, plus root of degree n plus 1 of n minus 1 , end of root, ; BLEU = 0.36, WER1 = 1.0222, WER2 = 1.0189;

4. Subscripts, special characters and summation:

$$\sum_{k=1}^N z_k \cdot \bar{z}_k = \sum_{k=1}^N (a_k + b_k \cdot i) \cdot (a_k - b_k \cdot i) = \sum_{k=0}^{N-1} (a_{k+1}^2 + b_{k+1}^2)$$

- Reference 1: summation from k equals 1 to N of z sub k, dot z bar sub k, equals summation from k equals 1 to N of open parenthesis a sub k, plus b sub k, dot i closed parenthesis dot open parenthesis a sub k, minus b sub k, dot i closed parenthesis equals summation from k equals 0 to N minus 1 of open parenthesis a sub k plus 1, squared plus b sub k plus 1, squared closed parenthesis
- Reference 2: summation from k equals 1 to N of product of z sub k and z bar sub k, equals summation from k equals 1 to N of open parenthesis a sub k, plus b sub k, dot i closed parenthesis dot open parenthesis a sub k, minus b sub k, dot i closed parenthesis equals summation from k equals 0 to N minus 1 of open parenthesis a sub k plus 1, squared plus b sub k plus 1, squared closed parenthesis
- OM: summation from k equals 1 to N of z sub k, cdot z with bar sub k, end of summation equals summation from k equals 1 to N of open parenthesis a sub k, plus b sub k, cdot i closed parenthesis cdot open parenthesis a sub k, minus b sub k, cdot i closed parenthesis end of summation equals summation from k equals 0 to N minus 1 of open parenthesis a sub k plus 1, squared plus b sub k plus 1, squared closed parenthesis end of summation ; BLEU = 0.7514, WER1 = 0.1591, WER2 = 0.191;
- SM: summation from k equals 1 to N of z sub k, cdot z with bar sub k, end of summation equals summation from k equals 1 to N of open parenthesis a sub k, plus b sub k, cdot i closed parenthesis cdot open parenthesis a sub k, minus b sub k,

cdot i closed parenthesis end of summation equals summation from k equals 0 to N minus 1 of open parenthesis a subscript: k plus 1 end of subscript, squared plus b subscript: k plus 1 end of subscript, squared closed parenthesis end of summation ; BLEU = 0.6095, WER1 = 0.2727, WER2 = 0.3034;

$$\det A = \sum_{\sigma \in S_n} (-1)^{\text{sgn}(\sigma)} \prod_{i=1}^n A_{i,\sigma(i)}$$

- Reference: *det A equals summation over sigma in S sub n, of minus 1 to the power of: sgn of sigma end of exponent, product from i equals 1 to n of A sub i, sigma of i*
- OM: *det A equals summation over sigma in S sub n, of open parenthesis minus 1 closed parenthesis to the power of: sgn open parenthesis sigma closed parenthesis end of exponent, product from i equals 1 to n of A sub i, sigma of i, end of product end of summation*; BLEU = 0.6246, WER = 0.3659;
- SM: *det A equals summation over sigma in S sub n, of open parenthesis minus 1 closed parenthesis to the power of exponent sgn open parenthesis sigma closed parenthesis end of exponent product from i equals 1 to n of A subscript: i, sigma of i end of subscript, end of product end of summation*; BLEU = 0.466, WER = 0.5366 ;

5. Miscellaneous:

$$f(x) = \frac{1}{3x} \cdot \sin 2x$$

- Reference 1: *f of x equals one over 3x times sine of 2x*
- Reference 2: *f of x equals the product of one over 3x and sine of 2x*
- OM: *f of x equals one over 3x cdot sine of 2x*; BLEU = 0.7017, WER1 = 0.0909, WER2 = 0.2857 ;
- SM: *f of x equals fraction: numerator: 1, denominator: 3x, end of fraction cdot sine of 2x*; BLEU = 0.1751, WER1 = 1.1818, WER2 = 0.9286 ;

$$(n-2)! = \frac{n!}{n \cdot (n-1)}$$

- Reference 1: *n-2 factorial equals n factorial over n dot n-1*
- Reference 2: *n-2 factorial equals n factorial over n times n-1*
- Reference 3: *open parenthesis n-2 closed parenthesis factorial equals n factorial over n times open parenthesis n-1 closed parenthesis*
- OM: *open parenthesis n minus 2 closed parenthesis factorial equals fraction: numerator: n factorial, denominator: n cdot open parenthesis n minus 1 closed parenthesis, end of fraction*; BLEU = 0.1246, WER1 = 2.8889, WER2 = 2.8889, WER3 = 1.1765
- SM: *open parenthesis n minus 2 closed parenthesis factorial equals fraction: numerator: n factorial, denominator: n cdot open parenthesis n minus 1 closed parenthesis, end of fraction*; BLEU = 0.1246, WER1 = 2.8889, WER2 = 2.8889, WER3 = 1.1765

$$\int_2^{3\sqrt{2}} (x+1)^5 dx$$

- Reference 1: *integral from 2 to 4 cube roots of 2 of open parenthesis x plus 1 closed parenthesis to the fifth, dx*
- Reference 2: *integral from 2 to 4 cube roots of 2 of the quantity x plus 1, to the fifth, dx*
- OM: *integral from 2 to 4 cube root of 2 , end of root, of open parenthesis x plus 1 closed parenthesis to the fifth, dx*; BLEU = 0.6544, WER1 = 0.2727, WER2 = 0.4762 ;
- SM: *integral from 2 to 4 cube root of 2 , end of root, of open parenthesis x plus 1 closed parenthesis to the power of exponent 5 end of exponent dx*; BLEU = 0.4575, WER1 = 0.5909, WER2 = 0.8095;

Both metrics used for the algorithm evaluation prove that the Optimized Mode produced translations, which are much more consistent with the natural language when comparing to standard mode. The drawback of both natural language and OM is however, that it may cause confusion if not pronounced properly, i.e. with proper intonation indicating pauses. The main difficulty is the composition of the meaning of mathematical expression in a linear form, suitable to be spoken by TTS and understandable without ambiguity. To overcome this difficulty in the proposed translation rule set the commas were implemented as separators for the TTS system to properly indicate pauses. This solution worked well with the use of MSSpeech TTS engine built in Windows operating system, so we have tested it thoroughly with the voice "MS Paulina Desktop", which is available for free. The quality of sound was near to human and we did not experience any problems with understanding of individual words or intonation. In contrary, an open-source solution producing artificial speech called eSpeak was much less understandable even during the basic experiments [8].

It is worth to note, that mathematical notation has some special characteristics that make it sensitive to even small differences in translation algorithms. Most of all, the syntactical meaning of the expression strongly depends on single and sometimes very small symbols [39]. This sometimes happens in the other natural languages, but for proper verbalization of formal maths equations it seems to be essential. On the other hand, there are examples of translations which differ significantly due to the semantic interpretation done by the human reference reader. We are also aware of some abbreviations and possible ambiguities in the reference equations, which make them useful for the common e-learning purposes, but not for visually impaired audience. In general, resolving ambiguities in processing of mathematical notation is the challenging task and key issue to avoid computational overload [43]. Therefore, we should strive for very high overall results of quality factors in the math-to-speech field.

In Table 3 all obtained (best) translation quality characteristics are presented collectively. Clearly the OM shows better performance in resembling the natural spoken references than the SM Mode, both at average and in each of the particular examples. On the other hand, some rules used in SM mode (although commonly leading to the longer output) auto-protect against misunderstandings and ambiguities. Therefore it is not clear, which mode would better fit particular audience. Our system gives the choice at this point, helping to adapt the verbalization process to user's level of the experience.

4.4 User experience evaluation for common script editing tasks

To evaluate expansion possibilities of the developed translation system, we have conducted the survey about it among maths teachers. The participants had four tasks to solve, from the very basic one, to the more significant modifications of the translation function.

Table 3 Translation quality measures in different modes

Expression	WER OM	BLEU OM	WER SM	BLEU SM
$\frac{n-1}{n+1} + 1\frac{1}{2} + 5\frac{23}{3} - \frac{8}{4} \dots$	0.0238	0.9370	1.6667	0.2133
$2^i + 2^{i+1} + 2^k + 2^{3^2}$	0.1667	0.8132	0.4722	0.4354
$\sqrt{x} + \sqrt[3]{x-1} + \sqrt[4]{\frac{2}{5}} + \dots$	0.6640	0.5248	1.0222	0.3600
$\sum_{k=1}^N z_k \cdot \bar{z}_k$	0.1910	0.7514	0.3034	0.6095
$\det A = \sum_{\sigma \in S_n} (-1)^{sgn(\sigma)} \dots$	0.3659	0.6246	0.5366	0.4660
$f(x) = \frac{1}{3x} \cdot \sin 2x$	0.2857	0.7017	1.1818	0.1751
$(n-2)! = \frac{n!}{n \cdot (n-1)}$	2.8889	0.1246	2.8889	0.1246
$3\sqrt{2} \int (x+1)^5 dx$	0.4762	0.6544	0.5909	0.4575
Mean	0.6596	0.6415	1.0828	0.3552
Standard deviation	0.8566	0.2274	0.8013	0.1580

Solutions of these tasks were representative to common needs in the area of the translation of the mathematical notation into the speech. Knowledge about their quantitative (duration) and qualitative (difficulty) characteristics would certainly help to develop better and more efficient human-computer interaction in the field of the math-to-speech verbalization.

The tasks were as follows:

1. add root with explicit index equal 2 to the condition for the shortened version of root description in Polish,
2. add ordinals for two-digit exponents in Polish,
3. add information about definite integrals when both bounds are not empty,
4. add information about the end of integral for inner expressions with spoken transcription longer than 50 characters (normally math-to-speech system do not indicate end of integral, as it ends up with a symbol dx or its equivalent).

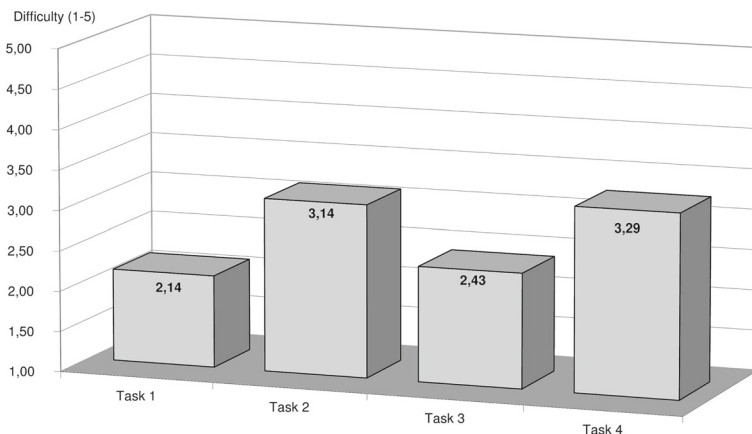


Fig. 3 The difficulty of typical translation tasks (average users scores)

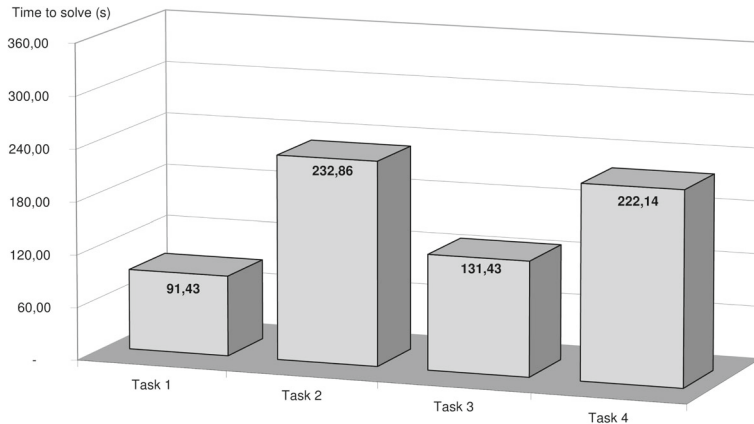


Fig. 4 The average time needed to solve typical translation tasks

The solution of the first task required just one additional condition, i.e. “`or (trim(indH) == '2')`” in appropriate translation function. To solve second task one should extend the helper function `IntToSpeech` and change the condition limits in the translation function. Third task required inserting two extra conditionals to `if` statement, so it was similar to the first one. In the last task the solution involved additional `if` statement with more complex condition than before, as well as modification of the result value.

Carefully conducted, all four tasks were not very difficult for participants, as all the teachers had at least a basic knowledge of programming. With the help of Lua syntax documentation they were able to extend existing translation functions or introduce additional conditions for the size and contents of the input data.

There were seven participants in our tests, everyone holding a university degree in mathematics or related disciplines. All of the respondents performed all the tasks, although three needed some help from the test moderator during experiment. The main test was preceded by some training conducted to ensure that participants understand the naming structure for Lua functions and the basics of the language itself.

The respondents rated the difficulty of all the tasks in a five point Likert scale: 1 – Very easy, 2 – Easy, 3 – Neutral, 4 – Difficult, 5 – Very difficult. In average, all tasks were rated as easy or neutral. The execution time of tasks correspond to the difficulty ratings, as illustrated in the Figs. 3 and 4.

Our experiment proved that introduction of scripting language affects the efficiency of translation in the positive way. The operator is able to modify, extend or correct the translation algorithm in a reasonable time. Moreover, it is possible to develop completely new translations for different national languages or mathematical dialects, all without the need to recompile the main application. We did not observe any significant problems in the usage of Lua language, as its syntax is simple and similar to Pascal, widely used in the education area.

5 Conclusions

The Lua scripts have proved to be very efficient in adapting our math-to-speech verbalizing system to various needs. The script code is quite intuitive, and makes the scripts easily editable to most users with mathematical background. The survey proved that the users are able to operate successfully with designed framework and are satisfied with the results.

Also, we introduced complex set of equations, covering all the details supported in the "Equation wizard" engine as a unit-test for the translation. This way we can ensure the quality of verbalization during changes in the script sources and avoid regression errors, which is extremely important from the engineering point of view.

The efficiency of the math-to-speech translation was evaluated with the use of standard metrics. It is worth noting that the analyzed equations were not trivial and contained nested sub-expressions, special Greek symbols and other diacritical marks as accents. The results of the evaluation of the reference translations based on English literature sources are quite promising, therefore we can hope for as good results of further studies involving native speakers. Moreover, the obtained results can be the solid base for the development of speech recognition systems complementary to math-to-speech interfaces.

Contrary to the various existing math-to-speech solutions based on the predefined hard-coded translation rules, our system offers a unique and novel possibility of adjusting the translation rules by the user. Our main contribution is the design of solution that allows to manually adjust the output translations to particular language and to follow the domain specific requirements (fields of mathematics, physics, etc.). This way the developed system becomes a versatile tool for various e-learning and assistive applications.

Declarations of interest: none.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Ahmetovic D, Armano T, Bernareggi C, Berra M, Capietto A, Coriasco S, Murru N, Ruighi A, Taranto E (2018) Axxessibility: a latex package for mathematical formulae accessibility in pdf documents. In: Proceedings of the 20th International ACM SIGACCESS conference on computers and accessibility. ACM, pp 352–354
2. Attanayake D, Denholm-Price J, Hunter G, Pfluegel E (2012) A Wigmore, Intelligent assistive interfaces for editing mathematics. In: Intelligent Environments (Workshops), pp 286–297
3. Attanayake D, Denholm-Price J, Hunter G, Pfluegel E (2014) Talkmaths over the web—a web-based speech interface to assist disabled people with mathematics. Proceedings of the Institute of Acoustics 36(36):443–450
4. Attanayake D, Denholm-Price J, Hunter G, Pfluegel E, Wigmore A (2015) Speech interfaces for mathematics: Opportunities and limitations for visually impaired learners. In: IMA international conference on barriers and enablers to learning maths: Enhancing learning and teaching for all learners
5. Attanayake D, Hunter G, Denholm-Price J, Pfluegel E (2013) Novel multi-modal tools to enhance disabled and distance learners' experience of mathematics. ICTer, vol 6, no 1
6. Balint L (1989) The role of models in handling complexity, flexibility and reliability of human-computer interfaces. In: Proceedings. VLSI and computer peripherals. COMPEURO 89, pp 2/118–2/120
7. Bateman A, Zhao OK, Bajcsy AV, Jennings MC, Toth BN, Cohen AJ, Horton EL, Khattar A, Kuo RS, Lee FA, Lim MK, Migasiuk LW, Renganathan R, Zhang A, Oliveira MA (2018) A user-centered design and analysis of an electrostatic haptic touchscreen system for students with visual impairments. Int J Hum Comput Stud 109:102–111. <https://www.sciencedirect.com/science/article/pii/S1071581917301301>
8. Bier A, Sroczyński Z (2015) Adaptive math-to-speech interface. In: Proceedings of the multimedia, interaction, design and innovation, ser. MIDI '15. ACM, New York, pp 7:1–7:9. <https://doi.org/10.1145/2814464.2814471>

9. Bocconi S, Dini S, Ferlino L, Martinoli C, Ott M (2007) ICT educational tools and visually impaired students: Different answers to different accessibility needs. Springer, Berlin, pp 491–500. https://doi.org/10.1007/978-3-540-73283-9_55
10. Caky P, Boron J, Klimo M, Bachrata K (2009) Mathematical formulas in text to speech system, Scientifi Letters of the University of Zilina, no. 3
11. Chang LA (1983) Handbook for spoken mathematics (Larry's Speakeasy). Lawrence Livermore Laboratory The Regents of the University of California
12. Corn A, Wall R (2002) Access to multimedia presentations for students with visual impairments. Journal of Visual Impairment and Blindness 96(4):197–211
13. Cuartero-Olivera J, Hunter G, Pérez-Navarro A (2012) Reading and writing mathematical notation in e-learning environments. eLearn Center Research Paper Series 4:11–20
14. Dobosz K (2016) Designing mobile applications for visually impaired people, Visually Impaired: Assistive Technologies, Challenges and Coping Strategies, pp 103–126
15. Dobosz K, Mazgaj M (2017) Typing braille code in the air with the leap motion controller. In: International conference on man–machine interactions. Springer, pp 43–51
16. Dobosz K, Szuścik M (2017) Onehandbraille: An alternative virtual keyboard for blind people. In: International conference on man–machine interactions, Springer, pp 62–71
17. Ehara T (2010) Machine translation for patent documents combining rule-based translation and statistical post-editing. In: NTCIR, pp 384–386
18. El-Glaly YN, Quek F (2014) Digital reading support for the blind by multimodal interaction. In: Proceedings of the 16th international conference on multimodal interaction, ser. ICMI '14. ACM, New York, pp 439–446. <https://doi.org/10.1145/2663204.2663266>
19. Fateman R (2013) How can we speak math? Computer Science Division, EECS Department, University of California at Berkeley, Tech Rep.
20. Ferreira H, Freitas D (2005) Audiomath: Towards automatic readings of mathematical expressions. In: Human computer interaction international (HCII). Citeseer, Las Vegas
21. Fong J, Ng M, Kwan I, Tam M (2003) Effective E-learning by use of HCI and web-based workflow approach. Berlin, Springer, pp 271–286. https://doi.org/10.1007/978-3-540-45200-3_26
22. Frankel L, Brownstein B, Soiffer N, Hansen E (2016) Development and initial evaluation of the clearspeak style for automated speaking of algebra. ETS Research Report Series 2016(2):1–43
23. Gawande V (2009) Effective use of hci in e-learning. In: Proceedings of the 6th international conference on eLearning for knowledge-based society eLearningAP 2009, December 17–18, 2009, vol 17. Bangkok Metropolitan Area, Thailand.
24. Isaac M, Pfluegel E, Hunter G, Denholm-Price J, Attanayake D, Coter G (2016) Improving automatic speech recognition for mobile learning of mathematics through incremental parsing. In: Intelligent environments (Workshops), pp 217–226
25. Kohanova I (2008) The ways of teaching mathematics to visually impaired students. In: International Congress on mathematical education (ICME)
26. Lewis C HCI for people with cognitive disabilities. In: ACM SIGACCESS accessibility and computing, 09 2005, pp 12–17
27. Líška M, Sojka P, Růžička M (2014) Math indexer and searcher web interface. In: International conference on intelligent computer mathematics, Springer, pp 444–448
28. Maćkowski M, Brzoza P, Żabka M, Spinczyk D (2017) Multimedia platform for mathematics' interactive learning accessible to blind people. Multimedia of Tools and Application, pp 1–18
29. Nazemi A, Murray I (2013) Mathematical formula recognition and transformation to a linear format suitable for vocalization. Int J Comput Sci Eng 5(9):847–855
30. Nazemi A, Murray I, Mohammadi N (2012) Mathspeak: An audio method for presenting mathematical formulae to blind students. In: 2012 5th international conference on human system interactions. IEEE, pp 48–52
31. Neto R, Fonseca N Camera reading for blind people, Procedia Technology, vol 16, pp 1200–1209, 2014, CENTERIS 2014 - Conference on ENTERprise Information Systems / ProjMAN 2014 - International Conference on Project MANagement / HCIST 2014 - International Conference on Health and Social Care Information Systems and Technologies. <https://www.sciencedirect.com/science/article/pii/S2212017314003624>
32. Papineni K, Roukos S, Ward T, Zhu WJ (2002) BLEU: a method for automatic evaluation of machine translation, pp 311–318
33. Połap D, Kęsik K, Książek K, Woźniak M (2017) Obstacle detection as a safety alert in augmented reality models by the use of deep learning techniques. Sensors 17(12):2803

34. Połap D, Woźniak M (2016) Introduction to the model of the active assistance system for elder and disabled people, In: International conference on information and software technologies, Springer, pp 392–403
35. Rezaei YA, Heisenberg G, Heiden W (2014) User interface design for disabled people under the influence of time, efficiency and costs. Springer International Publishing, Cham, pp 197–202. https://doi.org/10.1007/978-3-319-07854-0_35
36. Riga P, Kouroupetroglou G, Ioannidou P-P (2016) An evaluation methodology of math-to-speech in non-English DAISY digital talking books. In: International conference on computers helping people with special needs. Springer, pp 27–34
37. Salamonczyk A, Brzostek-Pawlowska J (2015) Translation of MathML formulas to Polish text, example applications in teaching the blind. In: 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF). IEEE, pp 240–244
38. Samsonov PA, Schöning J, Hecht B (2015) A user interface for encoding space usage rules expressed in natural language. In: Proceedings of the 33rd annual ACM conference extended abstracts on human factors in computing systems, ser. CHI EA '15. ACM, New York, pp 2211–2216. <https://doi.org/10.1145/2702613.2732737>
39. Sancho-Vinuesa T, Córcoles C, Huertas M, Pérez-Navarro A, Marquès D, Eixarch R, Villalonga J (2009) Automatic verbalization of mathematical formulae for web-based learning resources in an on-line environment, INTED2009 Proceedings, pp 4312–4321
40. Sheikh W, Schleppebach D, Leas D (2018) Mathspeak: a non-ambiguous language for audio rendering of mathml. *Int J Learn Technol* 13(1):3–25
41. Soldevila M, Ziliani B, Silvestre B, Fridlender D, Mascarenhas F (2017) Decoding lua: Formal semantics for the developer and the semanticist. arXiv:1706.02400
42. Spinczyk D, Maćkowski M., Kempa W, Rojewska K (2019) Factors influencing the process of learning mathematics among visually impaired and blind people. *Comput Biol Med* 104:1–9
43. Sroczyński Z (2010) Priority levels and heuristic rules in the structural recognition of mathematical formulae. *Theor Appl Inf* 22(4):273
44. Sroczyński Z (2014) Human-computer interaction on mobile devices with the fm application platform. In: Rostański M, Pikiewicz P (eds) Internet in the information society. Insights on the information systems, structures and applications. University of Dąbrowa Górnicza Press, pp 93–106
45. Sroczyński Z (2017) Actiontracking for multi-platform mobile applications. In: Computer science on-line conference, Springer, pp 339–348
46. Su W, Cai C, Wu J (2018) The accessibility of mathematical formulas for the visually impaired in china. In: International conference on artificial intelligence and symbolic computation. Springer, pp 237–242
47. Szopa R, Zuziak D (1994) Selected texts on higher mathematics, ser. Academic Textbooks, no 1837. SUT Press
48. Wigmore AM, Hunter GJ, Pflügel E, Denholm-Price J (2009) Talkmaths: A speech user interface for dictating mathematical expressions into electronic documents. In: International workshop on speech and language technology in education
49. Wolk K, Marasek K (2014) Real-time statistical speech translation. In: Rocha Á, Correia MA, Tan BF, Stroetmann AK (eds) New perspectives in information systems and technologies, vol 1. Springer International Publishing, Cham, pp 107–113. https://doi.org/10.1007/978-3-319-05951-8_11
50. Wongkia W, Naruedomkul K, Cercone N (2012) i-math: Automatic math reader for thai blind and visually impaired students. *Computers & Mathematics with Applications* 64(6):2128–2140



Agnieszka Bier is an assistant professor at the Faculty of Applied Mathematics of Silesian University of Technology, Gliwice, Poland. She received Ph.D. degree in mathematics from University of Silesia, Katowice, Poland in 2010 and MSc Eng in computer science from Silesian University of Technology in 2008. Her research interests include group theory, cryptography and computer assisted learning.



Zdzisław Sroczyński is a graduate at the Faculty of Automatic Control, Electronics and Computer Science. He received Ph.D. degree in computer science in 2011 from Silesian University of Technology, Gliwice, Poland. Currently an assistant professor at the Faculty of Applied Mathematics of Silesian University of Technology. His scientific interest include image and natural language processing, human-computer interaction, multimedia systems, assistive technologies, e-learning, programming of mobile devices.