Using NLG for speech synthesis of mathematical sentences

Alessandro Mazzei
Università degli Studi di Torino
alessandro.mazzei@unito.it

Michele Monticone
Università degli Studi di Torino
michele.monticone@edu.unito.it

Cristian Bernareggi
Università degli Studi di Torino
cristian.bernareggi@google.com

Abstract

People with sight impairments can access to a mathematical expression by using its LaTeX source. However, this mechanisms have several drawbacks: (1) it assumes the knowledge of the LaTeX, (2) it is slow, since LaTeX is verbose and (3) it is error-prone since LaTeX is a typographical language. In this paper we study the design of a natural language generation system for producing a *mathematical sentence*, i.e. a natural language sentence expressing the semantics of a mathematical expression. Moreover, we describe the main results of a first human based evaluation experiment of the system for Italian language.

1 Introduction

The recent progress of computational linguistic techniques and frameworks had a deep impact in the field of the assistive technologies. For instance, the recent development of commercial platforms for building speech dialogue systems, which are designed for not-impaired people, can also help people with disabilities in daily activities. For example a vocal command can be used to unlock a door in a house. However, for more specialized activities one needs to understand the necessity of specific communities in specific domains.

In the case of mathematical domain, blind people can access to a mathematical expression by using its LaTeX source. However, this process have several drawbacks. First of all, it assumes the knowledge of the LaTeX. Second, listening LaTeX is slow, since LaTeX is verbose. Finally, it is error-prone since LaTeX is a typographical language, that is a language designed for specifying the details of typographical visualization rather than for efficiently communicate the semantics of a mathematical expression. For instance, the simple LaTeX expression $\$f(x)\$$ is just a typographical description and so it represents both the function application of f to x, and the multiplication of the constant f for the constant x surrounded by parenthesis.

In this paper we study the design of a natural language generation (NLG) system for producing a *mathematical sentence*, that is a natural language sentence containing the semantics of a mathematical expression. Indeed, humans, when have to orally communicate mathematical expressions, use their most sophisticated communication technology, that is natural language. However, with respect to other domains, the mathematical domain has a number of peculiarities for speech that needed to be accounted for (see Section 4).

We have three main research goals in this paper. The first goal is answering to the question: *what is the linguistic status of a mathematical expression?* In other words, we want to investigate about the possibility to use the standard notions of linguistics, primarily syntax, for mathematical sentences. The second goal concerns the *possibility to use a standard NLG architecture*, that is a sentence planner and a realizer, for the production of a mathematical sentence. The third goal concerns *the possibility to simplify the listening of a mathematical expression by using speech features* during the speech synthesis. Indeed, in contrast with other fields, the mathematical domain is essentially a spoken domain (Chang, 1983). Indeed, by only listening the audio format of mathematical sentence, that is without accessing to its written form, the standard precedence of the mathematical operators are hardly recognizable. In other words, speech features, as pauses and prosody, can modify the perceived structure of the mathematical sentence in a peculiar way.

The schematic architecture of the developed framework is designed in Figure 1. The schema follows the well-known approach of the *interlingua* of rule-based machine translation (Hutchins

Proceedings of The 12th International Conference on Natural Language Generation, pages 463–472,
Tokyo, Japan, 28 Oct - 1 Nov, 2019.

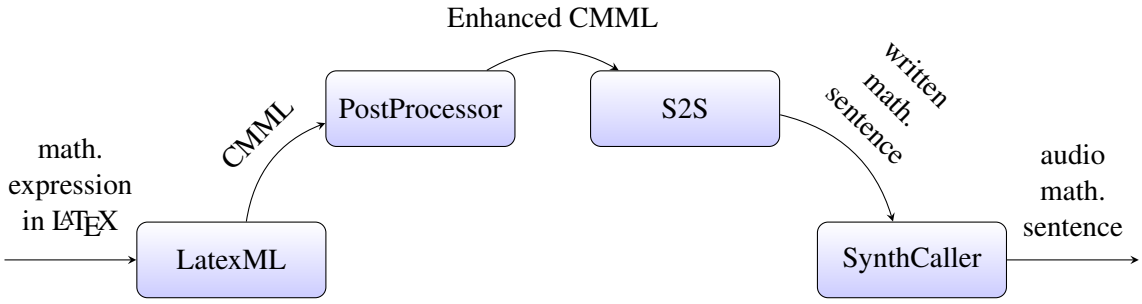

Figure 1: The software architecture for the generation of mathematical sentences. The information flow starts from (1) the LaTeX representation of the expression, (2) its translation in CMML, (3) enhancement of CMML, (4) generation of the written form of the mathematical sentence, (5) production of the audio form of the mathematical sentence.

and Somer, 1992). The process of generating a mathematical expression from its LaTeX source is a two-step algorithm. In the first step the LaTeX is analyzed and its semantics is represented in *Content MathML* (CMML henceforth), a W3C standard[1] for the syntax and semantics of mathematical expressions. In the second step, the CMML representation is used as input of the S2S (Semantics to Speech) module, that is a NLG module, to generate the mathematical sentence and, after the introduction of parenthesis or pauses, its audio format encoding.

The paper is structured as follows. In Section 2 we give a short review of the accessibility problem of mathematical expressions for visually impaired people. In Section 3 we describe the first step of the algorithm, that is the process to extract the CMML representation from its LaTeX representation. In Section 4 we describe our assumptions about the syntactic structures associated to mathematical operators. In Section 5 we describe the second step of the algorithm, that is the NLG of the mathematical expression from its CMML representation. In Section 6 we describe a first human-based evaluation of the system for Italian language performed by four blind people. Finally, Section 7 closes the paper with some considerations and pointing to future work.

2 Related work

Research to enable people with sight impairments to access mathematical notation has been conducted in two main directions. On one hand, different techniques have been investigated to preserve mathematical notation in a source format, that can be processed by a screen reader, throughout the whole workflow of a scientific document. In particular, nowadays it is possible to embed mathematical expressions in web pages not only as images, which cannot be processed by screen readers, but through MathML or MathJax (Cervone, 2012) and in PDF documents produced from LaTeX through the LaTeX package Axessibility (Ahmetovic et al., 2018). On the other side, many research works have investigated how people with sight impairments can read and understand mathematical notation, along two directions: conversion into Braille and speech reading. Since Braille is not a universal standard, different converters have been developed. The most widespread include conversion from LaTeX to Japanese Braille (Hara et al., 2000), to Nemeth code mostly used in English speaking countries (Papasalouros and Tsolomitis, 2015), to Marburg code mostly used in German speaking countries (Murillo-Morales et al., 2016) and from MathML to Spanish, French and Italian Braille codes (Soiffer, 2016). Nonetheless, Braille cannot support all the notations that can be expressed through LaTeX or presentation MathML (e.g., category theory, computational logic) and it has not a mechanism to introduce new notations hence the converters have a number of limitations. For what concerns speech reading, different techniques have been investigated. First, the most common approach transforms LaTeX or MathML expressions into a readable sentence by mapping a sequence of mathematical symbols to an aural equivalent for English (Raman, 1996), Spanish, German and French (Soiffer, 2007), Polish (Bier and Sroczynski, 2015) and Thai (Boonprakong et al., 2017). This approach is totally unambiguous, but it is very verbose (e.g. multiple nested parentheses can be hardly retained). Moreover, only a limited number of mathematical contexts are managed. Second, in addition to se-

[1] https://www.w3.org/TR/MathML3/chapter4.html

```
<m:apply>
   <m:eq/>
   <m:ci>y</m:ci>
   <m:apply>
      <m:times/>
      <m:ci>f</m:ci>
      <m:ci>x</m:ci>
   </m:apply>
</m:apply>
```

```
<apply>
   <eq/>
   <ci>y</ci>
   <apply>
      <ci>f</ci>
      <ci>x</ci>
   </apply>
</apply>
```

Figure 2: On the left the CMML generated by LatexML. On the rigth the enhanced CMML obtained after the preprocessing phase and .

quential speech reading, hierarchical exploration of the mathematical expression is provided (Soiffer, 2007; Sorge et al., 2014). This approach reduces the mental workload to retain the chunks of the expression. Third, speech reading is generated in a controlled environment such as a mathematical editor (Waltraud Schweikhardt, 2006; Raman and Gries, 1994). The context is defined by the author, hence the speech reading can be more accurate with respect to the semantics.

The idea to use mathematical sentences for improving the accessibility of mathematical expressions has been previously presented and experimented in (Ferres and Fuentes Sepúlveda, 2011; Fuentes Sepúlveda and Ferres, 2012) for Spanish language. However, in contrast to (Ferres and Fuentes Sepúlveda, 2011; Fuentes Sepúlveda and Ferres, 2012), we use a linguistic-based NLG architecture rather than a template-based one. In particular, by using the SimpleNLG realization engine for Italian, we allow both (i) for portability of the system to other languages, and (ii) a major and simple customization of the mathematical sentences. Indeed, the linguistic nature of the SimpleNLG input format allow for a very simple implementation of linguistic operations, as coordination or punctuation insertion (e.g. parenthesis. Cf. (van Deemter et al., 2005) for a discussion on the advantages and limitations of the template-based approach).

3 From LaTeX to CMML

The first step of our algorithm is the generation of CMML associated to a LaTeX formula. We based this step on an external tool named *LatexML* (Miller, 2007). However, the CMML obtained from this tool needed to be enhanced by a post-processing procedure for two distinct reasons.

1. We decided to clean the CMML obtained from LatexML since we wanted to remove some extra characters in the tag names (e.g. the suffix `m:`). With the same aim, we replaced all non standard characters from the tag values, e.g. some variable names which are written by the LatexML with italics font. Moreover, in certain cases LatexML generate some tags with the *open math* standard, as the case of `conditional-set`, that is converted by using the csymbol tag. For the sake of generality, in order to simplify the generation step of the system, we decided to uniform these cases to their corresponding *pure CMML* tag.

2. There are case in which the typographical origin of the LaTeX creates ambiguity that cannot always correctly solved by LatexML. If $\$y = f(x)\$$ is the LaTeX representation of formula $y = f(x)$, LatexML cannot autonomously decide the correct relation between the symbols f and x. One could force its interpretation as a function application (i.e. f is a function and x is his argument) or in alternative could force its interpretation as a multiplication (i.e. both f and x are just variables). Indeed, by default LatexML always assumes latter option. As a consequence, we had to fix it by hand in a number of cases.

The output of LatexML and the enhanced version with the hand fix applied are shown in Figure 2.

4 The (linguistic) Syntactic Structure of the Mathematical Expressions

Mathematical notation has been conceived with the aim of representing mathematical concepts using a specific written symbolic language. However, it is used in speech as *standard* language where usual syntactic notions, as number agreement, have to be accounted. For example the, ut-

Category	**Operators**	Construction
relational	$>, \geq, \gg, <, \leq, \ll, =, \neq, \subset, \not\subset, \subseteq, \not\subseteq$	Copula
arithmetic, algebraic, set	$+, -, *, /, \circ, [x]^{[n]}, \in, \notin, \cap, \cup, \setminus, \times$	Declarative
logical connectives	$\wedge, \vee, \neg, \implies, \iff$	Coordination
elementary functions	$\sin, \cos, \tan, \arcsin, \arccos, \arctan, \lvert \dots \rvert, \sqrt[n]{x}, n!, f^{-1}$	Noun Phrase
sequence	$\sum_{[x=a]}^{[b]} [f(x)], \prod_{[x=a]}^{[b]} [f(x)], \lim_{[x \to a]} [f(x)]$	Noun Phrase
calculus	$\int_{[a]}^{[b]} f(x)\, \mathrm{d}x, f^{(n)}(x), \frac{\mathrm{d}^n f(x)}{\mathrm{d}x^n}$	Noun Phrase
conditional set	$\{[vars] \mid conditions\}$	Reduced Relative
pair	$([x], [y])$	Reduced Relative

Table 1: In this table are shown all operators considered in our work, divided in category by the same linguistic structure.

terance $2/3$ can be pronounced as `two thirds`. Establishing what are the similarities between mathematical language and natural language is important for theoretical and practical considerations since this would imply the existence of linguistic structures which could be exploited for language generation. Also very simple mathematical expression can often be pronunced in different ways which underlay different mathematical sentences and so different syntactic structures, as $2/3$ that can also be pronunced with the sentence `two over three` (Chang, 1983). In contrast, there are cases in which the parsing of a mathematical expression unambiguously conducts to a specific mathematical sentence and to a unambiguous syntactic structure. For example, the mathematical expression $|x|$, that corresponds to the mathematical sentence *the absolute value of x* seems obviously to be a noun phrase modified both by an adjective and by a propositional phrase. In contrast, for the mathematical sentence `x belongs to A`, the most natural syntactic analysis seems to be a declarative sentence. However, there are many cases where the syntactic structure of a mathematical sentence is not so simple to represent. For example, `x plus three` could be analyzed as composed by the subject `x` and the object `three`, but this declarative representation of the sentence clashes with the impossibility to assign the part of speech verb to the word `plus` in the standard language.

As working hypothesis, we decided to assume a "specialized" syntactic analysis for a number of mathematical objects. For instance, `x plus three` indicates the action of adding one quantity to another, so it can be represented as a declarative structure. As a consequence, `plus` can be analysed as verb and this assumption can be extended to all the mathematical sentences.

In this paper we considered only the mathematical structures belonging to the subfield of the mathematical analysis. In particular we considered all the mathematical expressions in an Italian analysis book (Pandolfi, 2013). By using this corpus of expressions and by assuming that all numbers and variables can be treated as nouns and that all arithmetic operators can be treated as verbs, we found eight additional categories for representing all complex mathematical expressions and we defined a specific syntactic construction for each category. We reported these eight categories in Table 1.

We decided to analyse the mathematical sentences of relational operators as copula sentences (`a is greater than b`), algebraic operators as declarative sentences (`a cartesian product b`), logical operators as conjunctions (`a or b`), elementary operators (e.g. radical sign), sequence (e.g. limit), calculus (e.g. integral) as noun phrases (`the square root of x`), pairs and conditional sets as reduced relatives (`the set of x such that x is less than 3`). It is worth noting that our syntactic representations for mathematical operators in the analysis domain could have alternative representations or could be specialized in a more refined classification (c.f. (Chang, 1983)). However, we decided to use only eight category

for the sake of simplicity.

5 Building Mathematical Sentences with NLG

Traditional NLG architectures split the generation process into three distinct phases, that are *document planning*, *sentence planning* and *realization* (Reiter and Dale, 2000; Gatt and Krahmer, 2018). In particular document planning decides *what* to say and sentence planning and realization decides *how* to say it. In this project the content of the communication is specified by the input mathematical expression, so the content selection phase is not necessary at all.

In Section 5.1 we will give some details on the rule-based sentence planner designed for managing mathematical sentences and in Section 5.2 we will describe the use of the SimpleNLG-it realizer (Mazzei et al., 2016) for the case of mathematical domain.

5.1 Building a Sentence Planner for Mathematical Sentences

The input of the sentence planner is a mathematical expression in the form of enhanced CMML, that is a sort of linguistic semantic tree for the mathematical expression. In order to associate a *sentence plan*, that is a sort of under-specified tree-based syntactic structure, we devised a recursive algorithm that traverses top-down the CMML structure.

We classified all the mathematical expressions into a number of predefined categories, as discussed in the Section 4 (see Table 1). In particular, for each category we designed a prototypical sentence plan that will be used in the recursive process. Each prototype builds a specific linguistic construction (e.g. *copula, reduced relative* etc.), that is designed for giving syntactic roles to the arguments of the specific mathematical construction. For instance, on the left of the Figure 3, we reported the prototypical sentence plan for the *conditional set* mathematical structure and on the right of we reported an example of its instantiation. Note that in the produced structures: (1) the leaves of the sentence plan are lemmas rather than words, (2) the syntactic relations among the nodes are expressed using both dependency relations (e.g. subj, complement) as well as constituency nodes (e.g. Prepositional Phrase, PP). This peculiar representation is typical of some realization engines. and will be exploited in the next step of the generation.

In order to build a sentence plan for a mathematical sentence, there are two important issues: (i) the perception of precedence of the arithmetic operator and (ii) its most economical non ambiguous representation.

(i) Listening mathematics has some peculiarities with respect to reading it. For instance, division is granted a higher precedence than addition, and during the reading process the expression $a+b/c$ is parsed as $a+\frac{b}{c}$ without ambiguities. A different result arises if one listens the equivalent mathematical sentence `a plus b divided by c` without reading the expression: we experimented that the most frequent perceived parse is $\frac{a+b}{c}$. After a limited number of experiments in listening arithmetic expressions with distinct (blind and not blind) people, we decided to state as working hypothesis that *the precedence of the arithmetic operators are perceived in the reverse order when one listens a mathematical expressions without reading it*. We are aware that this speculation should be supported by specific experimental studies but, at the best of our knowledge, we have not been able to find them.

(ii) A different problem concerns the most efficient way to represent the *correct* precedence of a mathematical expression. In other words, how we can build a mathematical sentence unambiguously equivalent to $a + \frac{b}{c}$? A trivial but effective solution is to use parenthesis, that is to produce the mathematical sentence `a plus open parenthesis b divided by c close parenthesis`. However, the drawback of this solution is the length of the sentence that, for very complex expressions, can augment substantially.

In order to account for the problem of the precedence of the operators and its representation, we modified the sentence planner in two ways. First, we decided to model parenthesis as *first-class citizens* in the sentence plan, that is we considered *open-parenthesis* and *closed-parenthesis* as two new lexical items of the SimpleNLG lexicon which can be used as pre-modifier and post-modifier of a mathematical sentence respectively.

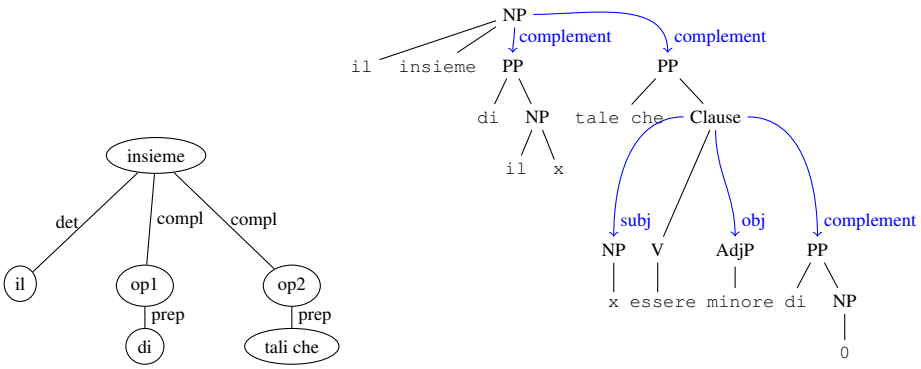

Figure 3: The prototypical sentence plan for the *conditional set* mathematical structure (left), and its fulfillment producing the sentence `L'insieme degli x tali che x è minore di 0` (rigth, *the set of all x such that x is lesser than 0*).

Second, similar to (Fuentes Sepúlveda and Ferres, 2012), we allowed to use a *speech pause* as a synonymous of open/closed-parenthesis items.

In order to experiment the pros and cons of using parentheses and pauses in the understanding of a mathematical sentence, we decided to implement three distinct parenthesization strategies, called *parenthesis*, *pause*, and *smart*.

1. In the *parenthesis strategy*, all the necessary parentheses are inserted in the sentence plan. Note that a parenthesis has to be considered necessary with respect to the inverted precedence order hypothesis stated above.

2. In the *pause strategy*, all the necessary pauses are inserted in the sentence plan.

3. In the *smart strategy*, all the necessary parentheses are inserted in the higher nodes of the sentence plan, and the necessary pauses are inserted close to the leaves of the sentence plan. This is a hybrid strategy that combines parentheses and pauses in order to have a less verbose mathematical sentence.

The evaluation of the performance of these parenthesization strategies is one of the goal of the experimentation described in the Section 6.

5.2 Using SimpleNLG for spoken mathematics

In order to produce a spoken mathematical sentences in Italian with the SimpleNLG-it realizer (Mazzei et al., 2016), we needed to account for the construction of a domain specific lexicon for the field of the mathematical analysis.

SimpleNLG-it is the Italian porting of the SimpleNLG realizer, that was originally designed only for English (Gatt and Reiter, 2009). As default Italian lexicon, SimpleNLG-it uses a basic vocabulary of around 7000 words, that is a *simple* lexicon studied to be perfectly understood by most Italian people (Mazzei, 2016). However, for this specific project we needed to augment the basic lexicon with a mathematical specialized lexicon, that contains both new lexical entries (as `arcotangente`, *arctangent*), than new associated value for lexical entry that are yet in the basic lexicon (as the *noun* value for the part of speech of the lemma `integrale`, *integral*). This specialized lexicon contains 113 entries that are mostly categorized as nouns (e.g. `logaritmo`, *logarithm*), verbs (e.g. `intersecare`, *intersect*), adjective (e.g. `iperbolico`, *hyperbolic*). In the lexicon, there are only two new instances of adverbs (that are `relativamente` and `propriamente`, *relative, properly*), and only one instance of "prepositional locution" (that is `tale che`, *such that*). Finally, we added specific lexical items to realize both parenthesis (that are `parentesi aperta` and `parentesi chiusa`, *open/closed parenthesis*) and speech pause. This latter item will be finally realized by using the SSML (Speech Synthesis Markup Language) tag `<break/>`, that can be processed by many speech synthesis engines[2]. An example of written mathematical sentence generated for the mathematical expression $\sqrt[n]{x} = x^{1/n}$ is `<break time="1000ms"/>La radice n-esima di x è uguale a x elevato a <break time="500ms"/> 1 diviso n` (*the n-th root of x is equal to x raised to 1 divided by n*).

[2] https://www.w3.org/TR/speech-synthesis11/

Simple formulas	Complex Formula
$A \times B = \{(x, y) \mid x \in A, y \in B\}$	$\lim_{x \to x_0} \left\{ \frac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right\} = 0$
$g^{-1}(y) = f^{-1}\left((y-b)/a\right)$	$y = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$
$\int_b^c a \, \mathrm{d}x = a(c - b)$	$\int \frac{1}{\sqrt{m^2 - x^2}} \mathrm{d}x = \arcsin \frac{x}{m} + c$
$x > b \Longrightarrow \lvert f(x) \rvert < M$	$\sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$
$\sqrt[n]{x} = x^{1/n}$	$\lim \left(1 + \frac{1}{n}\right)^n = e$

Table 2: On the left are shown the simple formulas. On the right the complex ones.

The actual version of the mathematical sentence generator has been interfaced with two speech synthesis engines, that are the web service provided by the IBM-Watson framework[3] (*W-engine* henceforth), and the Espeak API[4] (*E-engine* henceforth). W-engine is a commercial, closed software based on deep learning, while E-engine is a free, open-source software based on formant synthesis algorithms. Note that for not visual impaired people W-engine sounds more fluent but for visual impaired people sounds more familiar since it is used by the free NVDA screen reader.

6 Evaluation

In order to have a first evaluation of the generation system, we wanted to test the hypothesis that the system produces mathematical sentences which are understandable by visually impaired people. So, we built a web-based test explicitly designed for this class of users. We designed a questionnaire composed by a 6 multiple choices questions concerning personal data, a core of 25 open questions each one concerning the listening of a mathematical sentence and its comprehensibility, 1 Likert-scale question globally comparing LaTeX and system comprehensibility, 1 open question for free comments.

The 25 core questions have a all the same schema: there is a audio file encoding a mathematical sentence and there is a open form for transcribing it. In the compilation instructions, we asked the users to fill this section by using "LaTeX or with other non ambiguous formal representation". The mathematical expressions obtained have been manually translated to CMML for evaluation. We implemented the questionnaire by using the Google Form framework, that was preliminarily judged accessible by a blind person.

We have built the 25 core questions by using 10 mathematical expressions belonging to the analysis book used for developing the generation module (Pandolfi, 2013) (see Table 2). We have selected simple 5 expressions (less than 15 nodes in their *MathML* representation) and 5 complex expressions (more than 15 nodes). By varying (i) the synthesis engines (W-engine and E-engine), and (ii) the parenthesization strategies (*parentheses*, *pauses* and *smart*), we instantiate 25 possible mathematical sentences of the 10 mathematical expressions in the questionnaire.

In order to score the comprehension of the user we used two distinct metrics on the CMML expressions, that are *Exact Match* and *SPICE* (Anderson et al., 2016) metrics. The exact match returns 1 (or 0) value if the starting CMML tree and the perceived CMML are equals (or not). The SPICE score is a tree similarity measure previously used in the context of automatic caption generation. SPICE is obtained by computing the F-score of the overlap between two trees: the overlap is measured by decomposing trees in typed elementary substructures, that are operands, operators and their relations. For instance, the expression $x - 1$ is decomposed as $\{1, x, \textit{minus}, (\text{op: } \textit{minus}, \text{first: } x), (\text{op: } \textit{minus}, \text{second: } 1)\}$ (cf. (Anderson et al., 2016) for more details).

For the experimentation, we recruited 4 visually impaired people with personal invitation without any rewards. All users are Italian mother tongue, have a good knowledge of mathematical analysis and have a bachelor degree (only one related to mathematics). We believe that for a preliminary evaluation of the system, 4 blind people is a valu-

[3] https://www.ibm.com/watson/services/text-to-speech/

[4] http://espeak.sourceforge.net

able number. Indeed, note that all the users found the complete experimentation quite tiring and they spend around one hour to complete it. Note that in previous work on the generation mathematical sentences (Ferres and Fuentes Sepúlveda, 2011; Fuentes Sepúlveda and Ferres, 2012), the evaluations have been performed without blind people[5]. So, in our knowledge, this is the first NLG study on mathematical sentence with a realistic users' test-set.

We also submitted the questionnaire to a not visually impaired analysis teacher. She found the task of understanding mathematical sentence only by listening extremely hard. We did not report her (very low) scores since it is outside of the main goal of the experimentation, but this poor results show the different perception difficulty among different kind of users.

6.1 Results and Error Analysis

In Table 3 and in Table 4 are reported the scores obtained by the users for Exact Match and SPICE measures. From the first column of Table 3

U	All (25)	Simple (7)	Comp. (18)
1	13	5	8
2	18	5	13
3	13	5	8
4	14	5	9
Tot.	58 (58%)	20 (71%)	38 (53%)

Table 3: The Exact Match measure for all mathematical sentences (25 instances), only simple (7 instances), only complex (18 instances).

U	All (25)	Simple (7)	Comp. (18)
1	0.92 (0.11)	0.95 (0.1)	0.91 (0.11)
2	0.96 (0.06)	0.98 (0.05)	0.97 (0.07)
3	0.93 (0.09)	0.90 (0.14)	0.94 (0.06)
4	0.95 (0.07)	0.97 (0.09)	0.94 (0.07)
Avg.	0.94 (0.08)	0.95 (0.01)	0.94 (0.08)

Table 4: The averaged SPICE measure and standard deviations for all mathematical sentences (25 instances), only simple (7 instances), only complex (18 instances).

we can see that score variation seems strongly depend by the user. Indeed, User 2 is the only one with a Master Degree related to mathematics. A possible comparison can be done with (Fuentes Sepúlveda and Ferres, 2012), where a user group of 20 not-blind engineers have been exposed to 15 mathematical sentences generated from mathematical formulas obtained from Wikipedia. (Fuentes Sepúlveda and Ferres, 2012) obtained a final score for the exact match between $76 - 79\%$, that is comparable with the results of Table 3.

The SPICE scores in Table 4 confirms the dependence from the user of the results and evidences that only small portions of the mathematical expressions have been misunderstood. A manual inspection of the results showed that most of misunderstood sentence have structural errors in the perceived tree (11 errors), while are less frequent erroneous symbols (6 errors). For instance, a common mistake regarded the perception of the derivation operator, that is `la derivata di f di x` (*the derivative of f of x*).

By considering the Simple and Complex columns of the Table 3 is evident that, not surprisingly, the complexity of the expression have a strong impact on the comprehension and the t-test returns a value of 0.01 (two-tailed p-value). However, the application of t-test to the SPICE values for Simple and Complex expression do not consider them significantly different (0.75 two-tailed p-value).

In Table 5 we reported the averaged values of SPICE for three distinct categorizations of the mathematical expressions, that are parenthesization strategy, synthesis engine and *fame*. This last category expresses that the last two complex expressions in Table 2 are two well-known mathematical definitions and this fact could affect their comprehension.

With respect to the main goal of searching for specific strategy for simplify the listening of a mathematical expression, we note that the statistical analysis of the values in Table 5 concerning category parenthesization did not suggest any significant variations among the strategies. For instance, by comparing with t-test the SPICE scores of parenthesis and pauses strategies, we obtain a not significant value (0.75 two-tailed p-value). In contrast we could assert as posthoc hypotheses that both synthesis engine and fame have a significant effect on the performance of the system: by applying the t-test we obtained for both 0.02 two-tailed p-value. However, new experiments with more users are necessary to confirm these conclu-

[5](Ferres and Fuentes Sepúlveda, 2011) used automatic evaluation for the coverage and (Fuentes Sepúlveda and Ferres, 2012) recruited "20 engineers and engineering students" for the correctness.

Cat.	U1	U2	U3	U4
Par.	0.87 (0.16)	0.98 (0.06)	0.93 (0.06)	0.93 (0.08)
Pause	0.95 (0.04)	0.95 (0.11)	0.96 (0.04)	0.96 (0.04)
Smart	0.93 (0.10)	0.98 (0.04)	0.92 (0.11)	0.95 (0.09)
W-eng.	0.91 (0.11)	0.97 (0.07)	0.92 (0.09)	0.94 (0.08)
E-eng.	0.99 (0.03)	0.99 (0.03)	0.97 (0.04)	0.97 (0.04)
Famous	1.00 (0.00)	0.96 (0.10)	1.00 (0.00)	1.00 (0.00)
NFamous	0.89 (0.11)	0.97 (0.05)	0.90 (0.09)	0.93 (0.08)

Table 5: The averaged SPICE measures and standard deviations for different categorizations of the mathematical expressions.

sions.

7 Conclusion

In this paper we have presented a study on the generation of mathematical sentences, i.e. sentences in natural language expressing mathematical expressions[6].

We have described the peculiarities of the mathematical domain with respect to the task of NLG. In particular, we have considered the practical problem of analysing the semantics expressed by a mathematical expression in LaTeX, and its speech production in Italian language by using the SimpleNLG-it realizer and two distinct synthesis engines. We have proposed three different strategies for parenthesization of ambiguous mathematical expressions based on both parenthesis and pauses in speech. Finally, we have conducted a human-based evaluation of the system by using a tree-based measures of similarity. The results of the experimentation suggests (1) a preference for formant-based synthesizer with respect to NN-based synthesizer, (2) the performances on simple expressions are good, but (3) on more structurally complex expressions need improvements

In future work we intend to expand the lexicon in order to use English language and so to perform evaluation with a larger number of users and with more complex procedures of parenthesization. In particular, we want to experiment the use of acoustic signals for represents the opening and the closure of nested parentheses. Moreover, we intend to integrate the system into a dialogue system architecture with the idea to allow user to ask for repetition and clarification in the case of very complex expressions.

[6]The open-source licensed software can be downloaded at https://bitbucket.org/tesimagistralemonticone/formula-to-speech/

References

Dragan Ahmetovic, Tiziana Armano, Cristian Bernareggi, Michele Berra, Anna Capietto, Sandro Coriasco, Nadir Murru, Alice Ruighi, and Eugenia Taranto. 2018. Axessibility: A latex package for mathematical formulae accessibility in pdf documents. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '18, pages 352–354, New York, NY, USA. ACM.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: semantic propositional image caption evaluation. *CoRR*, abs/1607.08822.

Agnieszka Bier and Zdzislaw Sroczynski. 2015. Adaptive math-to-speech interface. In *Proceedings of the Mulitimedia, Interaction, Design and Innnovation*, MIDI '15, pages 7:1–7:9, New York, NY, USA. ACM.

Nattapat Boonprakong, Patcharida Pudpadee, Thanarat H. Chalidabhongse, and Proadpran Punyabukkana. 2017. Reading mathematical expression in thai. In *Proceedings of the 11th International Convention on Rehabilitation Engineering and Assistive Technology*, i-CREATe 2017, pages 9:1–9:3, Kaki Bukit TechPark II,, Singapore. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

Davide Cervone. 2012. Mathjax: A platform for mathematics on the web. *Notices of the American Mathematical Society*, 59.

Lawrence A. Chang. 1983. Handbook for spoken mathematics (larry's speakeasy). *Lawrence Livermore Laboratory, The Regents of the University of California.*

Leo Ferres and José Fuentes Sepúlveda. 2011. Improving accessibility to mathematical formulas: the wikipedia math accessor. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility, W4A 2011, Hyderabad, Andhra Pradesh, India, March 28-29, 2011*, page 25.

José Fuentes Sepúlveda and Leo Ferres. 2012. Improving accessibility to mathematical formulas: The

wikipedia math accessor. *New Rev. Hypermedia Multimedia*, 18(3):183–204.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Albert Gatt and Ehud Reiter. 2009. SimpleNLG: A Realisation Engine for Practical Applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 90–93, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shunsuke Hara, Nobuyuki Ohtake, Mika Higuchi, Noriko Miyazaki, Ayako Watanabe, Kanako Kusunoki, and Hiroshi Sato. 2000. Mathbraille; a system to transform latex documents into braille. *SIGCAPH Comput. Phys. Handicap.*, (66):17–20.

W. John Hutchins and Harold L. Somer. 1992. *An Introduction to Machine Translation*. London: Academic Press.

Alessandro Mazzei. 2016. Building a computational lexicon by using SQL. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016.*, volume 1749, pages 1–5. CEUR-WS.org.

Alessandro Mazzei, Cristina Battaglino, and Cristina Bosco. 2016. SimpleNLG-IT: adapting SimpleNLG to Italian. In *Proceedings of the 9th International Natural Language Generation conference*, pages 184–192, Edinburgh, UK. Association for Computational Linguistics.

Bruce Miller. LaTeXML: A LaTeX to XML converter [online]. 2007.

Tomas Murillo-Morales, Klaus Miesenberger, and Reinhard Ruemer. 2016. A latex to braille conversion tool for creating accessible schoolbooks in austria. volume 9758, pages 397–400.

Luciano Pandolfi. 2013. *ANALISI MATEMATICA 1*. Dipartimento di Scienze Matematiche "Giuseppe Luigi Lagrange", Politecnico di Torino.

Andreas Papasalouros and Antonis Tsolomitis. 2015. A direct tex-to-braille transcribing method. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '15, pages 373–374, New York, NY, USA. ACM.

T. V. Raman. 1996. Emacspeak—direct speech access. In *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, Assets '96, pages 32–36, New York, NY, USA. ACM.

T. V. Raman and David Gries. 1994. Interactive audio documents. In *Proceedings of the First Annual ACM Conference on Assistive Technologies*, Assets '94, pages 62–68, New York, NY, USA. ACM.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Neil Soiffer. 2007. Mathplayer v2.1: Web-based math accessibility. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '07, pages 257–258, New York, NY, USA. ACM.

Neil Soiffer. 2016. A study of speech versus braille and large print of mathematical expresisons. In *Lecture Notes in Computer Science*, volume 9758, Berlin. Springer.

Volker Sorge, Charles Chen, T. V. Raman, and David Tseng. 2014. Towards making mathematics a first class citizen in general screen readers. In *Proceedings of the 11th Web for All Conference*, W4A '14, pages 40:1–40:10, New York, NY, USA. ACM.

Kees van Deemter, Emiel Krahmer, and Mariet Theune. 2005. Real versus template-based natural language generation: a false opposition? *Computational linguistics*, 31(1):15–23. Imported from HMI.

Nadine Jessel Benoit Encelle Margaret Gut Waltraud Schweikhardt, Cristian Bernareggi. 2006. Lambda: A european system to access mathematics with braille and audio synthesis. In *Lecture Notes in Computer Science*, volume 4061, Berlin. Springer.