Peter Caky – Juraj Boron – Martin Klimo – Katarina Bachrata *

MATHEMATICAL FORMULAS IN TEXT TO SPEECH SYSTEM

A modular based system for synthesis of a written text allows solving partial problems separately. If there is designed a logical structure with clear interfaces, it could be possible to prepare modules for variant processes by different members of the team, without necessity of the coordinated work. The paper deals with a module for engine reading of mathematical formulas in a Slovak text-to-speech system. The problem consists in the conversion of formulas to the written text in Slovak language and consequently additional conversion to the phonetic form. The main idea is to use the markup language LaTeX.

1. Introduction

The communication between computers and people as end-users is most frequently based on writing and reading. Computers can perform written instructions and the users can read the result of computer's proceeding on monitors or printing output. But there are many situations when people need vocal communication with computers. People could be concentrated on some activity and they are not able to read or write (e.g. driving, surgery major), or people could have some healthy problem, like weak eyesight or dyslexia problems. For such reasons there were created tools for speech recognition and speech synthesis. Computers are able to create artificial sound which is similar to real speech. The text to speech synthesizer (TTS) is a system for conversion of text documents to audio files. Such a synthesizer (TTS KIS) is developed in the Department of Information Networks at the Faculty of Management and Informatics at Zilina University. The TTS KIS system is a modular based system processing the text with modules, converts it to phonetic transcription and prepares it for audio processing.

The final step of conversion is a preparation of artificial speech composed of small blocks of natural speech. There are many properties which have to be keep to prevent the production of a synthetic machine sounded speech. Such a speech is produced with identical sounds. People cannot produce identical parts of speech. They are not able to say two identical vowels "A". Therefore, an important task for synthesis is the transcription of a text, e.g.: "OKNO SA POOTVORILO" in such a way. that every letter "O" sounds a little bit different. On the other hand, the database including all the words of the Slovak language (and all the different forms of every word) is very large. Contrary to the English language, here it is not possible to use the database with complete words. So, there is a database of diphones (two joined phones cut from an artificial word) and synthesized words are combined from these diphones. The phonetic transcription has to be prepared with specified properties for every word and sentence. The properties like melody, pronunciation, phrase breaks, volume, intensity, pitch must be described. These properties are controlled by separate modules in the TTS KIS system. Modules give instructions for the next process with the text and put them into phonetic transcription. The module described in this article deals with mathematic formulas and expressions. It translates formulas written in the LaTeX language into a phonetic form suitable for next processing.

The TLatex module which is responsible for reading mathematical formulas is designed and implemented into the TTS KIS real system and enhances the possibilities of the TTS synthesizer. In the paper we will describe general principles of modular processing of the text by using Speech Synthesis Markup Language (SSML). Then we will describe the LaTeX program as a markup language and will take advantage of their similarities.

2. Description of TTS KIS system

Speech synthesis is based on concatenation of diphones, the sound formed from two phones running from the center of the first phone to the center of the second one. In this sound a natural transition from the first phone to the second phone is recorded. Connection of the phones is realized in the centre of the phone, where the behavior of the phone as a time process is well predictable. The modular based program for text-to-speech uses a kernel developed by the team of researchers and students of the faculty. The database of diphones (approximately 2000 files) of the Slovak language was prepared for this system. The architecture of the system is based on SSML standards, Speech Synthesis Markup Language [1]. It is designed to provide a rich, XML-based markup language for assisting the generation of synthesis speech in Web and other applications. The important function of the markup lan-

* **Peter Caky, Juraj Boron, Martin Klimo, Katarina Bachrata**
Department of Information Networks, Faculty of Management and Informatics, University of Zilina, Slovakia, E-mail: peter.caky@gmail.com

guage is to provide a standard way to control properties of speech in different synthesis-capable platforms.

The input to the synthesis processor is a text prepared by a user or produced automatically. SSML standards define a form of the document. Next, there are few processing steps made by the synthesis processor to convert a marked-up text input into an automatically generated speech output. The markup language is designed for control over each of the steps described below and the document author can control the final voice output: XML parse, structure analyses, text normalization, text to phone conversion, prosody analyses and waveform production.

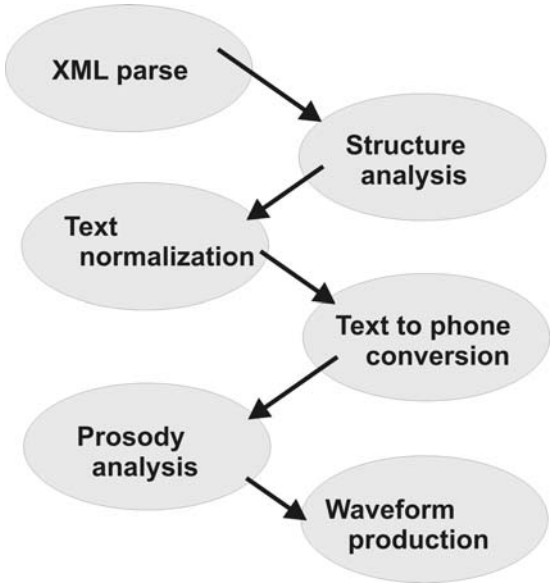

Fig. 1 Steps in the process of speech synthesis based on SSML

The marks in SSML language can be more or less structured. A very simple input to the synthesizer could have the following form:

```
<speak>
   Tento text ma byt precitany po slovensky.
Preto sme ho napisali po slovensky.
</speak>
```

The form of the input can be more complicated. The SSML is an annotation tool and it is possible to add next information. We can specify that between <speak> and </speak> elements there are two sentences. The input will be more specified:

```
<speak>
   <s> Tento text ma byt precitany po slovensky.
   </s>
    <s> Preto sme ho napisali po slovensky. </s>
</speak>
```

A user could continue and put next marks into the text for optimization the waveform output. After this processing the output will satisfy the user's conditions. The next example of a marked text illustrates a more detailed description of additional informational data:

```
<speak version”1.0” xml:lang=” en-US”>
   <lexicon
   uri="http://www.example.com/lexicon.file"/>
        <voice gender="female" variant="2">
          <p><s> Next text has to be read in
          Slovak language. </s></p>
          </voice>
</speak>
<speak version”1.0” xml:lang=”sk”>
        <voice gender="male" variant="1">
          <p><s> Tento text sme napisali po
          slovensky. </s></p>
          </voice>
</speak>
```

As shown the SSML gives possibility to set a lexicon, a voice, sentence boundaries that will be used for a grapheme to phonetic transcription. The annotations that SSML provides can be divided into three groups:

1. Document structure, text processing and pronunciation: <speak>, <language>, <lexicon>, <phoneme>, <say-as>, <p>, <s>.
2. Prosody and style: <voice>, <emphasis>, <break>, <prosody>.
3. Other elements: <audio>, <mark>, <desc>.

The requirements that appeared during the development of TTS KIS system lead to the extension of the SSML language to SSML+. This extension adapts the SSML to requirements for the TTS system which works with texts in the Slovak language. There are new elements and new attributes of the elements existing in the SSML+. It allows to structure texts in a special way typical for the Slovak language. Texts can be segmented more precisely to compound sentences, sentences, words, syllables and diphones. Due to this segmentation it is possible, for instance, to assign melody for particular sentences or words, which is important in Slovak pronunciation.

In order to achieve the complete SSML+ structure, there were defined and implemented modules for adding the necessary annotations into the input SSML text. Rule-based engines (RBE [2]) represent a way how this task can be accomplished. There was defined a set of atomically and independently executable modules. Each module M is determined by two sets of conditions:

1. Preconditions - a set of conditions that have to hold before a module code is executed
2. Postconditions - a set of conditions that must be true just after the execution of a module code

These modules are executed one by one in an undefined order depending on the result of module preconditions. The method of execution in which the order of module invocations is not explicitly defined is called the implicit invocation [3]. The order in which modules are executed is determined implicitly according to the

preconditions and postconditions and modules run till some precondition has changed.

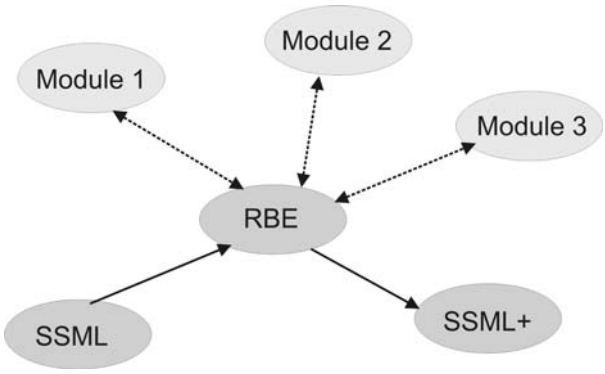

Fig. 2 Execution of modules based on preconditions

An advantage of the implicit invocation is its independence of the modules. Not only during the program running but also during the module development. The modules can be prepared separately. It is only necessary to define preconditions and postconditions for each module. Then it is easy to add the modules to the system. The TTS speech synthesizer uses a concatenative approach [4], [5] and some of the modules are used cause of the concatenative method:

1. Validation of the SSML and structure creation (speak, ssml, audio)
2. Structural analyses (lang, voice, paragraph, compound, sentence, sentence type, word, diphone, text norm, text, syllable, say as, sub, desc, mark, phoneme)
3. Grapheme to phoneme conversion (lexicon, ph, word join)
4. Prosody analyses (prosody, contours, range, pitch, duration, rate, volume, emphasis, voice, break)
5. Waveform production (diphone extract, diphone selection, diphone merge, contour application, prosody application)

But there are also modules for structural analysis and normalization of the text. The shortcuts, numbers, dates, marks are conversed into the text in these modules. We will focus on the Tcitac module[10] that executes mathematical expressions written in the LaTeX typesetting system.

3. Reading of mathematical expressions

From the synthesis point of view, a mathematical expression is a set of symbols, letters, numbers, formulas and relation signs which express some mathematical properties or data. Parts of sucha text could be numbers, variables, fractions, elementary functions, integrals, sums and whatever else that mathematicians can think up. People know the way how to read the expressions, they know where they have started and where the end of the expression is. If an expression is to be read automatically, people will have to teach computers how to converse a mathematical expression into a plain text. But there are many ways how to read the same expression also in the Slovak language.

A very easy example is the fraction 1/2. It is possible to say only “half”, but for the same expression we can use “one half”, “one over two”, “fraction one over two”, “fraction where numerator is one and denominator is two”. The most complicated way how to say 1/2 is the sentence:

> Begin of the mathematical expression start of the fraction start of the numerator one end of the numerator over start denominator two end of the denominator end of the fraction end of the mathematical expression.

People do not use the last way of reading the expression 1/2 but for complicated fractions this is a good solution how to do it. There are two choices in the Tcitac module for reading expressions: “simplified” reading (like “half”) and “normal” reading (as we could see above). The “simplified” reading is choice for reading texts with small numbers of expressions or for quick reading for first overview of the text. Technical articles require “normal” reading. The text must be read by separate steps. The next part of the text is read after finishing the previous part. If there are many levels in one part, they have to be read progressively from complicated to simpler parts. After reading the elementary part the system continues by reading on a higher level.

$$\lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k^2}=\frac{\pi^2}{6} \qquad (1)$$

There is a scheme of the way of “normal” reading of the formula (1) in Figure 3. At this scheme there are arrows denoting progress of the steps of reading blocks of an expression.

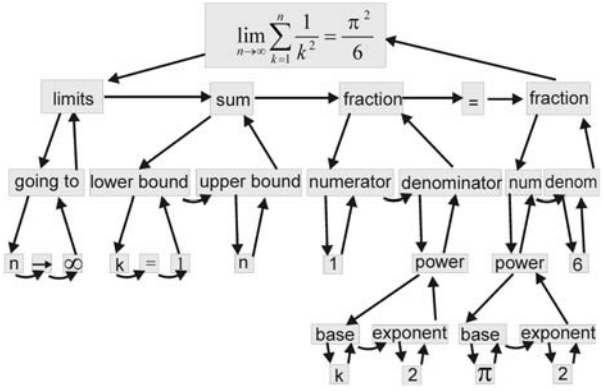

Fig. 3 Scheme of the steps in reading of mathematical formula

The final text for normal reading will be prepared in the form:

> Begin of expression limit for en going to infinity from sum for kej equal to one till en from fraction start of the numerator one end of the numerator over start denominator k power to two end of the denominator end of the sum end of the limits is equal to fraction start of the numerator pi power to two end of the numerator over start denominator six end of the denominator.

The "simplified" reading uses the same scheme of reading the expression. Short forms of the formulation are used. The final text for "simplified" reading will be prepared in the following form:

```
Limit for en going to infinity from sum for kej equal to one till en from one over square of is equal to square of pi over six.
```

A "simplified" form is easier to use but there can be some mistakes in understanding. A better solution is to use a combination of these two forms. The first "simplified" reading and then if there are some doubts about understanding the "normal" reading follows.

4. Mathematical expressions in LaTeX typesetting system

LaTeX is a markup language for preparing documents for the TeX typesetting program designed by Donald Knuth, [6], [7]. LaTeX is most widely used by mathematicians, scientists and academic society. The typesetting system offers extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies. LaTeX was originally written by Leslie Lamport [8]. Both of them LaTeX and TeX are open source systems. They say about LaTeX that it is a document preparation system for high-quality typesetting. LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content. One of the greatest motivating forces for Donald Knuth when he began developing the original TeX system was to create something that allowed a simple construction of mathematical formulas, whilst looking professional when printed. The expression (1) written in LaTeX can be seen in Figure 4.

$$\lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k^2}=\frac{\pi^2}{6}$$

Fig. 4 Output of mathematical formula written in LaTeX

A LaTeX document consists of one or more source files with extension .text and files which can be written in an optional text editor. In this file there is a text and commands commonly start with a backslash and are grouped with curly braces. We will focus on writing mathematical formulas in LaTeX. LaTeX needs to know beforehand that the subsequent text does in fact contain mathematical elements. This is because LaTeX typesets math notation differently than a normal text. Therefore, special environments have been declared for this purpose. For declaration special marks are used allowing to recognize mathematical formulas in the text. There are special marks in LaTeX for mathematical symbols, variables and characters. Formula (1) can be expressed as:

```
$$\lim\limits^{n\to\infty}\sum\limits_{k=1}^n\frac{1}{k^2}=\frac{{\pi}^2}{6} $$
```

Other ways how to declare mathematical environment are pairs of mark like $ ... $, \begin{math}... \end{math}, \begin{displaymath}... \end{displaymath}, \begin{equation}... \end{equation}. The LaTeX commands are sorted to several classes according to the way in which they are read:

- signs without special commands in mathematical environment (numbers, letters, + * > < ( ) [ ] : " ' /)
- pairs of the marks for mathematical environment ($...$, \begin{displaymath}... \end{displaymath}, \begin{equation}... \end{equation}, \begin{eqnarray}... \end{eqnarray}) and similarly pairs of the marks for norm, absolute value and curly brackets
- non reading commands like spaces, labels, size and style of the characters, commands for citing, index and references
- non parametric commands (\infty, \alpha, \%, \dots, n \choose k, \subset, \to, \into, \partial, \bot, \rightarrow, \forall, \neq, \&)
- standard commands for functions (\lim, \det, \min, \cos, arsin)
- letters of Greek alphabet (\alpha, \beta, \gamma, \epsilon, \varepsilon, \pi, \phi, \varphi)
- commands after parameter (\overline{x}, \overbrace{x}, \underline{x}, \dot{x})
- commands with parameters like exponent, index, root, limit, product, sum, integral, fraction, vectors (x^{2\pi}, x_{ij}, \sqrt[3]{9}, \lim_{n\to \infty}{\fraction 1 n}, \prod_{k=0}^{5} {x_{k}}, \sum_{k=0}^{\infty} {n^2}, \int_a^b x^2 dx , \frac1 5, \vec{v})

We will describe principles of the Tcitac module of the TTS KIS synthesizer in the next section.

5. Description of Tcitac module

The Tcitac is a module working with mathematical expressions written in LaTeX. The module converts mathematical expressions to the SSML+ format prepared for reading. More precisely, the pre-condition of this module is at least one appearance of LaTeX mathematical environment. The postcondition of the Tcitac is a phonetic transcription of all the LaTeX formulas. The module consists of nine classes shown in the UML diagram in Figure 5. The Class Texformula is an interface of the module. In this class we detect if the precondition of the module is satisfied. Class Ttexty contains commands with their phonetic transcriptions, class TNumbersTransformer converts numbers, class TTexTransformer finds the beginning of the formulas and creates an instance TProstredie. Tprostredie finds LaTeX commands and creates an instance TCast for conversion of individual commands. Tcast executes the command recursively and after finishing it it returns the control back to the upper level class TProstredie and the results of TProstredie are returned to TTexTransformer and then to Texformula. Texformula continues in scanning all documents and returns the changed SSML document and finishes its proceeding. The instance of class Tprikaz is used to transmit information about the current proceeding of translation of the command. The instance of class Tznak helps in sequential reading of the signs of the document.

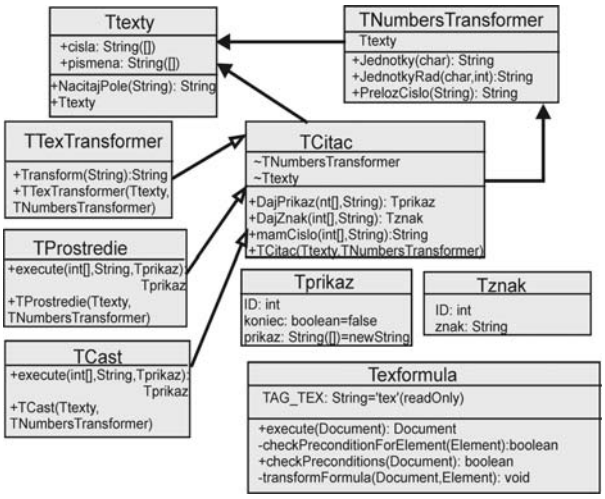

Fig. 5 UML diagram of the classes of module Tcitac

6. Conclusions

The Tcitac module does not have a fixed programmed list of the LaTex commands and a way of how to read them. The information about reading is loaded from the input files. It is easy to add new commands or to edit some existing ones. Possibilities of changes of the phonetic transcription are very important properties of the Tcitac module. There are many exceptions or different ways in reading mathematical formulas. The improvement of reading can be done by including exceptions to the lexicon of the TTS KIS system [11]. Exceptions and mistakes in the phonetic transcription can be included into the learning of neural networks. Works [12], [13] can be extended from reading the text to reading formulas and mathematical expressions.

References

[1] Speech Synthesis Markup Language (SSML) Version 1.0, http://www.w3.org/TR/speech-synthesis
[2] Rule Based Engine, http://en.wikipedia.org/wiki/Rules_engine
[3] AI Application Programming, M. Tim Jones, 2003 Charles River Media, ISBN 1584502789
[4] Improvements in Speech Synthesis: Cost 258: The Naturalness of Synthetic Speech, Eric Keller, 2002 John Wiley and Sons, ISBN 0471499854
[5] CAKY P., KLIMO M., MIHALIK I., MLADSIK R.: *Text, Speech and Dialogue 2004,* 7th International Conference, TSD 2004, Text to Speech for Slovak Language, pp. 291-298, ISBN 978-3-540-23049-6, 2004 Springer, ISBN 3540230491
[6] KNUTH, D. E.: *The TeXBook,* Massachusetts: Addison-Wesley, 1984), x+483pp. ISBN 0-201-13448-9
[7] KNUTH, D. E: *Typesetting Concrete Mathematics,* TUGboat 10 (1989), 31–36, 342. Reprinted as chapter 18 of Digital Typography.
[8] LAMPORT, L.: *LaTeX: A Document Preparation System,* Addison-Wesley. Retrieved on 2007-02-02, 1986.
[9] KLIMO, M., MIHALIK, I.: *Extending annotational SSML into structural content for speech synthesizer (in Slovak),* Scientific Papers of the University of Pardubice, 12 (2006): Series B-ISBN 978-80-7194-985-5, p. 193–199.
[10] NOVAK, A.: *Mathematical LaTeX Formulas Reading in TTS (in Slovak),* dipl. práca, 2007
[11] CAKY, P., KLIMO, M., MIHALIK, I., MLADSIK, R.: *Text-to-speech for Slovak language (in Slovak)* Proc. of Text, speech and dialogue: 7th international conference, TSD 2004, Brno, Czech Republic, 2004, Berlin: Springer, 2004, ISBN 3-540-23049-1, pp. 291–298.
[12] HUDEC, M.: *IT in Software Compensation Applications (in Slovak),* Ustav vedy a vyskumu UMB, 2006, Banska Bystrica ISBN 80-8083-196-3
[13] HUDEC, M.: Compensation Multi-voice Synthesis with Neuro-computing (in Slovak), Banska Bystrica, 2005, ISBN 80-8083-103-3