Web-Based System for Automatic Reading of Technical Documents for Vision Impaired Students*

Jindřich Matoušek[1], Zdeněk Hanzlíček[1], Michal Campr[2], Zdeněk Krňoul[1], Pavel Campr[1], and Martin Grůber[1]

[1] University of West Bohemia, Faculty of Applied Sciences, Dept. of Cybernetics, Univerzitní 8, 306 14 Plzeň, Czech Republic
[2] University of West Bohemia, Faculty of Applied Sciences, Dept. of Computer Science and Engineering, Univerzitní 8, 306 14 Plzeň, Czech Republic

Abstract. A web-based system for automatic reading of technical documents focused on vision-impaired primary-school students is presented in the paper. An overview of the system, both its backend (used by teachers to create and manage the documents) and frontend (used by students for viewing and reading the documents), is given. Text-to-speech synthesis utilised for the automatic reading and, especially, the automatic processing of mathematical and physical formulas are described as well.

Keywords: web-based system, automatic reading of technical documents, text-to-speech, reading of mathematical formulas, vision impaired.

1 Introduction

In this paper, a contribution to the integration of modern web with speech and language technologies is presented. More specifically, automatic reading of technical documents within the project ARET (Automatic Reading of Educational Texts for Vision Impaired Students) is introduced. The project aims at an innovation and enhancement of schooling of vision impaired (both purblind and blind) primary-school students and also at a facilitation of their self education. Technical documents include, but are not limited to, topics of Mathematics and Physics (ISCED 2 level).

Within the project a web-based system for automatic reading of technical documents was developed. Teachers use the system for a preparation, management and administration of educational texts. The texts are available to students online via system's frontend; they are read aloud by means of text-to-speech synthesis (TTS).

From the point of view of TTS, an automatic reading of technical documents and, especially, mathematical and physical formulas is very challenging, at least

* Project ARET CZ.1.07/1.2.00/08.0021 is co-funded by the European Social Fund and the State Budget of the Czech Republic. Support was also provided by the University of West Bohemia, project No. SGS-2010-054.

from two reasons. Firstly, text processing of formulas (i.e. decoding and transcription of their symbolic notation to a corresponding word form) is not trivial and requires a special treatment different from processing in a general-purpose TTS system. Secondly, most current TTS systems are based on a corpus-based approach to speech synthesis in that they are optimised on the corpora utilised during the system design. As mathematical and physical formulas are usually not included in these corpora, problems with synthetic speech quality (both intelligibility and naturalness) can arise when read automatically.

Speech- and language-processing technologies enable to develop assistive applications for people with various impairments or health disorders, see e.g. [1] for an example of a medicine application or [2] for an example of a system for deaf people. Some projects for reading technical documents or mathematical formulas for the vision impaired also exist. The problem of reading mathematics has been already solved, e.g. in the system AsTeR (Audio System for Technical Readings) [3] or in the system AudioMath developed at Porto University [4]. For the Czech language, the Lambda editor was created at Masaryk university (`http://www.teiresias.muni.cz/czbraille8`). Within the presented project ARET, a new system for reading mathematical formulas is being developed.

The paper is organised as follows. In Section 2, an overview of the developed system for automatic reading of technical documents is presented. The text-to-speech system used for reading the texts aloud is briefly described in Section 3. Special issues related to this specific utilisation of TTS technology are depicted in Section 4. Finally, conclusions are drawn in Section 5.

2 System Overview

The developed system is a web application, based on client-server architecture, running on *Apache* HTTP server with *MySQL* database system. The core of the system is based on *Symfony 1.4*, an open-source web application framework.

The system is divided into two separate sections: frontend and backend. Frontend serves as a public interface for viewing various technical documents (arranged as topics) and, at the same time, reading them. On the other hand, backend is an administrative interface, where the documents can be created and modified. A schematic view of the system can be seen in Figure 1.

2.1 Backend

System's backend is shown in Figure 2. Teachers have a direct access to the backend through a *WYSIWYG* text editor in which they create and modify the technical documents. Within the project ARET, the editor was enhanced with the ability to insert project-specific templates and formulas. The templates are used to clarify the meaning of a particular fragment of the document, for example the *Important* template is for highlighting a crucial information to which the students should pay more attention, or the *Example* template is used to display various examples. Different synthetic voices can be assigned to each template. Currently, five templates are supported: Definition, Important, Note, Example, and Solution.

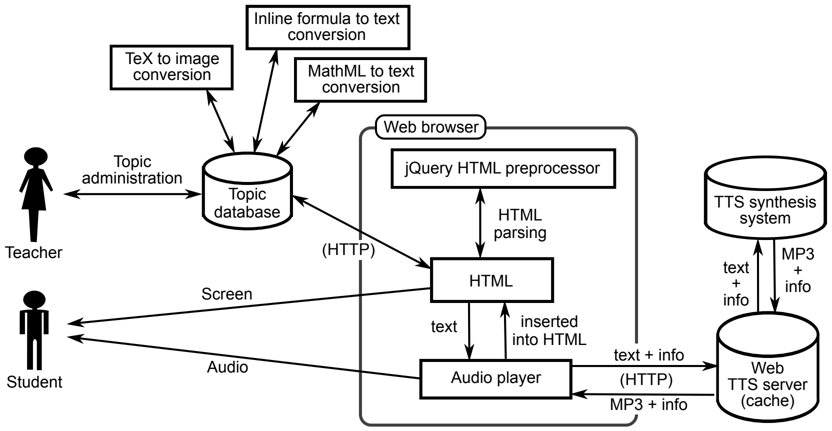
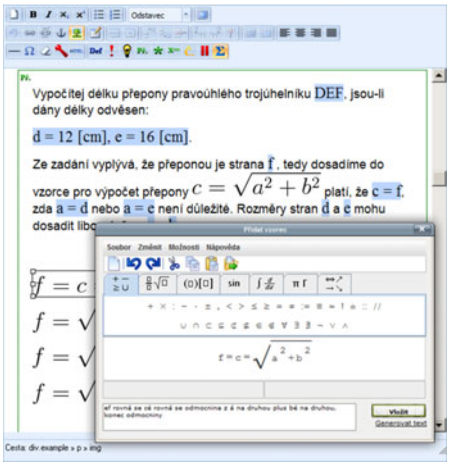


Fig. 1. System diagram



Vypočítej délku přepony pravoúhlého trojúhelníku DEF, jsou-li dány délky odvěsen:

d = 12 [cm], e = 16 [cm].

Ze zadání vyplývá, že přeponou je strana f, tedy dosadíme do vzorce pro výpočet přepony $c = \sqrt{a^2 + b^2}$ platí, že c = f, zda a = d nebo a = c není důležité. Rozměry stran d a c mohu dosadit libo

$f = c$

$f = \sqrt{}$

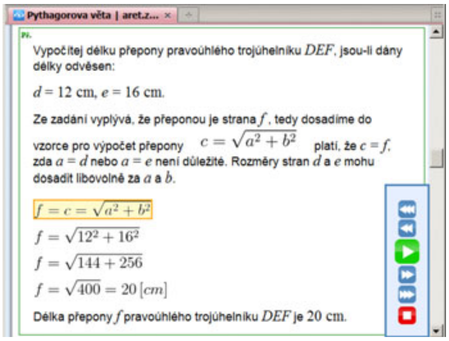
$f = \sqrt{}$

$f = \sqrt{}$

f=c=$\sqrt{a^2+b^2}$

ef rovná se cé rovná se odmocnina z á na druhou plus bé na druhou, konec odmocniny

Fig. 2. Backend – text editor and formula editor



Vypočítej délku přepony pravoúhlého trojúhelníku DEF, jsou-li dány délky odvěsen:

$d = 12$ cm, $e = 16$ cm.

Ze zadání vyplývá, že přeponou je strana f, tedy dosadíme do vzorce pro výpočet přepony $c = \sqrt{a^2 + b^2}$ platí, že $c = f$, zda $a = d$ nebo $a = e$ není důležité. Rozměry stran d a e mohu dosadit libovolně za a a b.

$f = c = \sqrt{a^2 + b^2}$

$f = \sqrt{12^2 + 16^2}$

$f = \sqrt{144 + 256}$

$f = \sqrt{400} = 20\,[cm]$

Délka přepony f pravoúhlého trojúhelníku DEF je 20 cm.

Fig. 3. Frontend – text which is currently being read is highlighted by the yellow colour; audio player interface is in the bottom right corner of the web page

There are two different ways for inserting mathematical formulas into the document. Simple formula with linear structure like $x + 1$ can be written as “inline formula”. To write more complex formulas, *DragMath* editor which enables to store a formula in both *MathML* and TEX formats (with MathML being used to derive a word-level description of the formula for speech synthesis—see Section 4—and with TEX being used to generate an image of the formula for displaying in the system’s frontend) was adopted.

2.2 Frontend

System's frontend is a public web interface where the topics are displayed and read aloud to students. Before displaying the web page, the HTML documents are automatically processed—optimised for TTS-based synthesising (the content texts, including the processed formulas, are extracted).

The texts are then sent to the *Web TTS server* which is responsible for the automatic reading. The text-to-speech conversion is made by the TTS system (described in Section 3) in which texts are synthesised as MP3 (or OGG) files. System's cache is also supported to avoid re-synthesising already synthesised texts. To play the audio files, JavaScript player using *Adobe Flash* or *HTML5 <audio> tag*, in which paragraph- and section-based navigation is supported, is employed.

3 Text-to-Speech

For the automatic reading of technical documents in system's frontend, Czech TTS system ARTIC [5] has been employed. ARTIC applies a corpus-based concatenative speech synthesis method. Based on a carefully designed speech corpus (a collection of a large number of utterances annotated on orthographic, phonetic and prosodic levels), statistical approach (with hidden Markov models, HMMs) was employed to perform an automatic phonetic segmentation of the source speech corpus into phones. Based on this segmentation, boundaries between diphones, the basic speech units used in the ARTIC system, were located. As a result, acoustic unit inventory (AUI), the source speech corpus indexed with diphones and prosodic structures, was built.

During run-time speech synthesis, phonetic and prosodic aspects of an input text are estimated first. Ideally, input text is a subject of a thorough analysis and processing. Due to a complexity of such a task, current text processing in the ARTIC system is somewhat simplified to four main steps: text normalisation of "non-standard" words (digits, abbreviations, acronyms, etc.) [6], detailed rule-based phonetic transcription, including pronunciation dictionary of "exceptional" words (mostly foreign words, names, physical units, etc.) [5], phones-to-diphones conversion, and prosodic description in terms of prosodic symbols (prosodic clauses, phrases, prosodemes, etc.) using prosodic phrase grammar [7]. Within the scope of the ARET project, text normalisation also includes the processing of mathematical and physical formulas described with the inline formulas or MathML codes. Such specially marked formulas could be then processed and converted to words (see Section 4).

Prosodic analysis includes punctuation-driven sentence clause detection, rule-based word stress detection and symbolic prosodic description [7]. Symbolic features based on a prosodic phrase grammar, like prosodic sentence, prosodic clause, prosodic phrase, prosodic word, and prosodeme, were used to describe prosodic characteristics and to express prosodic structure of to-be-synthesised texts.

The resulting speech is generated by a *unit-selection* algorithm [8]. Its principle is to smoothly concatenate (according to *join cost*) speech segments (diphones in our case), extracted from natural utterances using the automatically segmented boundaries, from large speech unit inventories according to phonetic and prosodic criteria (*target cost*) imposed by the synthesised utterance. As there are usually many instances of each speech segment, there is a need to select the optimal (with respect to both target and join costs) instances dynamically during synthesis run-time (using the unit-selection technique). To calculate the target cost, a prosodic structure of the to-be-synthesised utterance is estimated, and a comparison between prosodic symbolic features (plus some positional features, like position of a diphone in a prosodic word, and contextual factors like immediate left and right phone) in the utterance and in the unit inventory is carried out. Join cost is evaluated as a distance between spectral features and pitch around the concatenation point of two potentially neighbouring speech units. After selecting the optimal sequence of (diphone) speech segments, neither prosodic nor spectral modifications are made in the ARTIC system except for simple smoothing at concatenation points. To cope with high CPU power and memory cost typical for unit-selection systems, a computational optimisation was carried out as described in [9].

4 Automatic Reading of Formulas

Text processing is an important part of a TTS system. Generally, text processing in a TTS system depends on the type of texts that are likely to appear at the input of the system. In the ARET project, educational texts (currently the texts of Mathematics and Physics at ISCED 2 level—i.e. mathematical and physical formulas) are expected as an input of the TTS system.

Reading of mathematical formulas in the Czech language is a very complex task, especially if the problem is supposed to be solved generally, i.e. there is no limitation for the complexity of the equation structure. In fact, any final system will be naturally limited by the definition of expected mathematical operations, types of operands, etc. Nevertheless, the system should be simply extensible by additional definition of reading rules, e.g. for new operators.

Since Czech is an inflective language, the correct grammatical form of particular operands (numbers and variables) can be different in various mathematical contexts. For the inflection, methods described in [6] were employed.

As mentioned in Section 2, two different representation of mathematical formulas are employed in our system. Simple formulas with a linear structure can be written and stored as a simple text. For the creation of more complex mathematical expressions, the special editor DragMath is employed and their structure is represented by using an MathML format. In both cases, thanks to a special syntax or marking of the formulas in the HTML code, no detection of formulas is needed.

4.1 Reading of “Inline” Formulas Represented by a Plain Text

Formulas written by a text (“inline formulas”) have a simple linear mathematical structure—it is usually a sequence of operators and operands, which can be read in the same order as it is written. Mathematical priority of particular operation has no significance for the reading. All operands in the formula have to be inflected into the corresponding grammatical form, which is determined by the previous operator (the first operand is in its primal form). We define a simple transcription rule for each operator, which contains

- transcription for a given operator
- grammatical form for the following operand (case, number and gender)

By the sequential application of those rules and operand inflection (described in [6]), the whole formula can be read.

In the current version, only some basic operators and operand types are supported in the text representation, e.g. addition, subtraction, multiplication, division, brackets, superscript (power), subscript, numbers, variables and physical units. For formulas with other operators or with a more complex structure, MathML representation has to be employed.

4.2 Reading of Formulas Represented by MathML

MathML (`http://www.w3.org/TR/MathML3`) is an XML application for describing mathematical notation. It is capable to represent mathematical formulas of almost any structure and complexity. Moreover, the standard can be easily extended with new operators or operand types. For purposes of our project, we defined a special operand type for physical units, an operator for applying operations on both sides of equation, etc.

The transcription of formulas represented by MathML can be divided into several steps:

- hierarchical decomposition of a MathML code
- selecting suitable transcription rules for particular operator
- applying the selected rules (operator transcription and the corresponding inflection of related operands)

For each mathematical operation, several transcription rules can be defined. They differ by their activation conditions, i.e. in various mathematical contexts, for various values or types of operands, different transcription rules can be selected. For most operators, one basic rule and several additional rules for exceptional cases are defined.

Each transcription rule contains a text template for the resulting expression together with the corresponding grammatical form for each operand (case, number, gender, cardinal or ordinal form, etc.). Moreover, a type of the resulting expression is also defined because this expression could be an operator in the higher level of the hierarchical structure representing the whole formula.

An illustrative (incomplete) example of transcription rules for two operators—power and fraction (in YAML notation):

```
POWER:
- condition: { operand_2_type: [ number, variable ] }
  operands:
  - { type: cardinal, case: 1, number: S, gender: F }
  - { type: ordinal, case: 4, number: S, gender: F }
  template: "{operand_1} na {operand_2}"
  expr_type: expression

FRACTION:
- condition: { any_operand_type: [ fraction, fraction_expression ] }
  operands:
  - { type: cardinal, case: 1, number: S, gender: F }
  - { type: cardinal, case: 1, number: S, gender: F }
  expression: "složený zlomek, nad hlavní zlomkovou čarou je
             {operand_1}, pod hlavní zlomkovou čarou je {operand_2}"
  expr_type: fraction_expression

- condition: { any_operand_type: [ expression, function ] }
  operands:
  - { type: cardinal, case: 1, number: S, gender: F }
  - { type: cardinal, case: 1, number: S, gender: F }
  expression: "zlomek, v čitateli je {operand_1}, ve jmenovateli je {operand_2}"
  expr_type: fraction_expression

- condition: { operand_1_type: [ number, variable ],
               operand_2_type: [ number, variable ] }
  operands:
  - { type: cardinal, case: 1, number: S, gender: feminine }
  - { type: cardinal, case: 7, number: S, gender: feminine }
  expression: "{operand_1} lomeno {operand_2}"
  expr_type: fraction
```

An example of MathML representation and transcription of a given formula follows. Aforementioned rules are applied in this example.

Formula	**MathML representation**	**Transcription**
$\frac{x^n}{y^3}$	`<math xmlns="http://www.w3.org/1998/Math/MathML"> <mfrac> <mrow><msup> <mrow><mi>x</mi></mrow><mrow><mi>n</mi></mrow> </msup></mrow> <mrow><msup> <mrow><mi>y</mi></mrow><mrow><mi>3</mi></mrow> </msup></mrow> </mfrac> </math>`	zlomek, v čitateli je iks na entou, ve jmenovateli je ypsilon na třetí

Particular rules for each operator are ordered and their condition evaluated from most special to most general. The last rule has usually an empty condition; thus, it is applied always when none from preceding rules is selected. It is quite easy to define a new set of transcription rules or extend an existing one with rules for new mathematical operations or with additional rules for some rare linguistic exceptions.

The conversion of formulas to text is shown in Figure 1 in blocks *"Inline formula to text conversion"* and *"MathML to text conversion"*.

5 Conclusion and Future Work

Automatic reading of technical documents within the project ARET, a contribution to the integration of modern web with speech and language technologies, was presented in the paper. Since ARET focuses on Mathematics and Physics, automatic processing of mathematical and physical formulas was dealt with. Nevertheless, the system framework has been designed to be general and flexible enough to cover also other kinds of technical documents, including more advanced topics like tertiary level of mathematics, etc. Although the ARET project is still being worked on, the first technical documents are already available on `http://ucebnice.zcu.cz`.

Future work will be focused on three main areas. First, the number of topics will be continuously increasing in order to cover the intended goal of the ARET project. Second, system functionality is also planned to be enhanced. For instance, other rules for reading formulas will be added. We also plan to personalise the system for each user by allowing him/her to change the layout of web pages with topics (e.g. colours of fonts, templates, etc.). Finally, we will also try to make the developed system more compatible with other tools and systems that vision impaired use. For instance, we will unify keyboard shortcuts.

References

1. Hippmann, R., Dostalová, T., Zvarová, J., Nagy, M., Seydlová, M., Hanzlíček, P., Kříž, P., Šmídl, L., Trmal, J.: Voice-supported Electronic Health Record for Temporomandibular Joint Disorders. Methods Inf. Med. 49(2), 168–172 (2010)
2. Krňoul, Z., Kanis, J., Železný, M., Müller, L.: Czech Text-to-Sign Speech Synthesizer. In: Popescu-Belis, A., Renals, S., Bourlard, H. (eds.) MLMI 2007. LNCS, vol. 4892, pp. 180–191. Springer, Heidelberg (2008)
3. Raman, T.V.: Audio System for Technical Readings. Ph.D. Thesis, Cornell University, New York (1994)
4. Ferreira, H., Freitas, D.: Enhancing the Accessibility of Mathematics for Blind People: The AudioMath Project. In: Miesenberger, K., Klaus, J., Zagler, W.L., Burger, D. (eds.) ICCHP 2004. LNCS, vol. 3118, pp. 678–685. Springer, Heidelberg (2004)
5. Matoušek, J., Tihelka, D., Romportl, J.: Current State of Czech Text-to-Speech System ARTIC. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2006. LNCS (LNAI), vol. 4188, pp. 439–446. Springer, Heidelberg (2006)
6. Zelinka, J., Kanis, J., Müller, L.: Automatic Transcription of Numerals in Inflectional Languages. In: Matoušek, V., Mautner, P., Pavelka, T. (eds.) TSD 2005. LNCS (LNAI), vol. 3658, pp. 326–333. Springer, Heidelberg (2005)
7. Romportl, J.: Prosodic Phrases and Semantic Accents in Speech Corpus for Czech TTS Synthesis. In: Sojka, P., Horák, A., Kopeček, I., Pala, K. (eds.) TSD 2008. LNCS (LNAI), vol. 5246, pp. 493–500. Springer, Heidelberg (2008)
8. Tihelka, D., Matoušek, J.: Unit Selection and Its Relation to Symbolic Prosody: a New Approach. In: Proceedings of Interspeech, Pittsburgh, USA, pp. 2042–2045 (2006)
9. Tihelka, D., Kala, J., Matoušek, J.: Enhancements of Viterbi Search for Fast Unit Selection Synthesis. In: Proceedings of Interspeech, Makuhari, Japan, pp. 174–177 (2010)