A MARKUP LANGUAGE FOR TEXT-TO-SPEECH SYNTHESIS

Richard Sproat[1], *Paul Taylor*[2], *Michael Tanenblatt*[1], *Amy Isard*[2]

(1) Bell Laboratories, Lucent Technologies
700 Mountain Avenue, Room 2d-451
Murray Hill, NJ 07974, USA
(2) Centre for Speech Technology Research,
University of Edinburgh
80 South Bridge, Edinburgh, U.K

ABSTRACT

Text-to-speech synthesizers must process text, and therefore require some knowledge of text structure. While many TTS systems allow for user control by means of ad hoc 'escape sequences', there remains to date no adequate and generally agreed upon system-independent standard for marking up text for the purposes of synthesis. The present paper is a collaborative effort between two speech groups aimed at producing such a standard, in the form of an SGML-based markup language that we call STML — Spoken Text Markup Language. The primary purpose of this paper is not to present STML as a fait accompli, but rather to interest other TTS research groups to collaborate and contribute to the development of this standard.

1. INTRODUCTION

In order to be able to make most efficient use of computers in the processing of online text, it is necessary to have mechanisms for marking those features of the text that are deemed to be salient, but which might be difficult or impossible to automatically detect in a general way. Ideally, such *markup languages* should be standardized in that they incorporate generally agreed upon conventions, and abstract in that they represent properties of the text with only minimal regard to the particular way in which that text will be used.

A simple example will serve to illustrate the point. In plain ('ascii') text there is no universally agreed upon convention for delineating paragraph boundaries, though there are several popular methods. Programs that need to find paragraph boundaries (e.g., a text formatter) will thus need to be somewhat clever, and they probably cannot perform entirely without error. In the SGML-based [1] markup language developed under the auspices of the Text Encoding Initiative [2], standard mechanisms, namely the markup tags <p> … </p>, are used to delimit paragraphs, meaning that in a text that is fully marked up according to the TEI conventions, there can never be ambiguity as to the location of paragraph boundaries. At the same time, notice that this markup is abstract in that it specifies nothing at all about how a paragraph might actually appear on a printed page: a text-formatting program for TEI documents might adopt any of a number of strategies of indentation or spacing, and still be faithful to the markup.

Text-to-speech synthesizers, like text-formatting programs, must process text, and therefore require some knowledge of text structure: at a minimum TTS systems must either compute (or be told) where paragraph, sentence and phrase boundaries lie; but additional information, such as whether a particular element of text represents a table, or a title, can be relevant to how that element should be rendered. We will refer to this kind of information as *text description*. In addition, it is usually desirable to be able to control certain properties of the synthesizer which do not relate directly to the structure of the text, but rather to its interpretation by the system: such properties include the particular speaker chosen for a passage, or the pitch range of a particular utterance. This latter kind of information we shall denote *speaker directives*.

Many TTS allow for markup of speaker directives: for example the Bell Labs synthesizers have a relatively rich set of 'escape sequences' for marking up such features as pitch range, accent patterns, and speaker characteristics. But such markup schemes are invariably system specific, and often ad hoc. This is problematic for developers who use TTS systems: what they require is a simple and clean interface which allows control over the main aspects of synthesis, and which is a standard such that they can use the same markup with a range of TTS systems.

The present paper is a collaborative effort between two speech groups (Bell Labs and Edinburgh University) aimed at producing a single markup language for speech synthesis. Previously, each of the sites has worked in isolation on this problem: this paper presents the results of our combined work to produce what we hope will be a single standard.

STML (Spoken Text Markup Language) is a markup language based on the Standard Generalised Markup Language. As such it has a clearly defined syntax and grammar, and there are many readily available tools for parsing and processing text marked up in this way. Crucially, STML proposes tags for both text description and speaker directives, and these tags can thus control all relevant aspects of the synthesis operation.

STML is the successor markup language to the systems previously developed by the two groups. SSML (the Speech Synthesis Markup Language) was developed at Edinburgh and was the first attempt at a TTS markup language [3]. This system provided a rudimentary set of system-independent tags based on SGML. The Bell labs system involved the use of a rich set of escape sequences in the input text. STML supersedes both these systems by inheriting the SGML nature of SSML and the wide coverage of the Bell Labs tag set.

In both its generality and its coverage, STML is superior to the only extant industry standard for TTS markup, namely Microsoft's Speech Application Programmer's Interface (SAPI). First, SAPI's tag syntax is completely ad hoc. Secondly SAPI provides tags only for speaker directives, not for text description. One reason for this is undoubtedly the SAPI designers' somewhat limited view of the function of TTS: "An application should use text-to-speech only for short phrases or notifications, not for reading long passages of text" [4, page 6]. In fact TTS systems are routinely used in many applications to read long passages of text, and therefore some mechanisms for defining textual properties seem necessary.

2. SAMPLE TAGS AND THEIR INTERPRETATION

The intended scope of STML is best illustrated by example. Consider the marked-up text in Figure 1.. Note that this example contains only a small subset of the STML tags that have been developed to date. The <LANGUAGE> and <SPEAKER> tags set (obviously enough) the language and default speaker for that language: in a multi-lingual TTS system, these specifications would cause appropriate language- and speaker-specific tables (e.g. pronunciation rules, acoustic inventory elements) to be loaded. The <GENRE> tag allows one to set the type of text (plain prose, poetry, lists ...). If a TTS system supports a particular genre type, then it will have at least some models particular to that genre. For example, a list type might be specified as having a different intonation pattern from that of plain prose. The <DIV> tag specifies a particular text-genre-specific division: in the example, it is used to mark the end of list items, for example. <EMPH> tags are used to place emphasis on words. The tag <PHONETIC> marks the contained region of text as being a phonetic transcription in one of a predefined set of schemes; in this case the scheme is the one *native* to the (hypothetical) TTS system reading this text. The <DEFINE> tag is used to specify the lexical pronunciation of a word. Here a non-native phonetic scheme is used: it is up to the TTS system to map from a foreign scheme to the native one.

The <BOUND> tag marks a prosodic phrase boundary: in the example, a minor boundary is marked. It is possible to change the voice by using the <SPEAKER> tag where speakers are named according to convention. Some standard names are given, (e.g. "male1" and "male2") which are mapped onto the set of voices available on the local system.

While the example may seem verbose, the number of tags can be reduced by SGML markup minimisation. This facility removes the need for redundant tags, such that during parsing, a </LANGUAGE> tag implies the presence of a </SPEAKER> tag immediately before it if this isn't marked explicitly.

Note that while we have, for the purposes of this example, presumed a TTS system reading the marked up document from beginning to end, other ways of rendering the text are also consistent with the purpose of STML. Following [5], one can also view a TTS system as an audio *browser*. In browser mode, a TTS system would present information in a more 'top-down' manner, giving the user a high-level view of the structure of the document, and allowing the user to navigate to interesting sections of the document.

The tags described above are defined in terms of their logical function. For example the <EMPH> *word* </EMPH> construction is defined as placing emphasis on *word* rather than in terms of any direct processing instructions to the TTS system. While in general we expect a TTS system to realise emphasis by making use of intonation, duration and energy, the specifics of how this is actually done are not specified.

3. ISSUES IN TAG DESIGN

In theory, anything specifiable in the text which can give an instruction or description to a TTS system could be in a synthesis markup language. In the interests of producing a compact, powerful and coherent set of tags, we used two criteria to decide which tags should be included in STML. The first is that computer-literate non-synthesis experts should be able understand what the tags mean and should be able to achieve the effects they want without too much effort. The second is that the tags should be portable across platforms so that a variety of synthesis systems should be able to make use of the additional information given by the tags. As it happens, these criteria tend to work together and favour the same types of tags.

Users unfamiliar with intonation would be daunted by having to using a notation system such as the tonal part of ToBI when specifying markup for intonation; and as many TTS systems do not use these labels internally anyway they would not serve well as a system independent intonation specification. Rather we have used tags such as <EMPH> and <BOUND> to specify intonation. These are easier to understand and are more likely to be common to most TTS systems. <EMPH> and <BOUND> are "higher level" than say, H* and L%, and so have a downside in that they do not offer the same level of overall control as the lower level equivalents. We consider this to be an acceptable tradeoff, i.e. the benefits of ease of use and cross platform portability outweigh the

```
<!DOCTYPE STML SYSTEM>
<STML>
<LANGUAGE ID=ENGLISH>
<SPEAKER ID=male1>
<GENRE TYPE=plain>
This is an example of some STML text.  In this example, we see some
<EMPH>particular</EMPH> STML tags, including:
<GENRE TYPE=list>
Language specification <DIV TYPE=item>
Speaker specification <DIV TYPE=item>
Text type (genre) specifications <DIV TYPE=item>
<PHONETIC scheme=native> f&amp;n"etik </PHONETIC> specifications
phrase boundary <BOUND TYPE=minor> specifications
</GENRE>

<DEFINE WORD="Edinburgh" PRO="e 1 d i n b 2 r @@" scheme="cstr">
This text can be processed by the Bell labs and Edinburgh Text to
speech systems.

Here is an example of the system switching to German
</GENRE>
</SPEAKER>
</LANGUAGE>
<LANGUAGE ID=GERMAN>
<SPEAKER ID=female1>
<GENRE TYPE=plain>
Guten Tag, meine Damen und Herren.
</GENRE>
</SPEAKER>
</LANGUAGE>
</STML>
```

Figure 1. An example of STML markup.

loss of fine control.

The problem of which tags to include in a generalised markup scheme has been addressed in the text formatting domain where the distinction between the *physical* and *logical* aspects of a document's markup has proved a useful concept in producing system-independent markup languages. The logical structure of a document is usually described declaratively, and without reference to how the formatting program should actually realise the desired format. In this way, it is possible for systems which work in different ways to process the same document successfully. Often the finished article appears different if processed on different systems, but in some sense the different versions are essentially the same. This is the aim we have for STML whereby the synthesized speech need not sound identical on different systems, but should give the same basic impression to the user.

3.1. Speaking Style

It is not always easy to maintain the logical and physical distinction. A good example involves the case of different speaking voices. Given that the TTS system being used can speak in a number of voices, it is possible that the listener will have preferences over which voice to use and will instruct the TTS system accordingly. In such cases, it is inappropriate for the choice of speaker to be included in the document itself, it is much more suitable if can be set externally. However, in certain cases, it can be argued that the choice of which speaker to use is important to the structure of the document. Imagine for instance that someone wishes to have the text of a drama synthesized: it is obviously important that each character is assigned a voice and that the choice of voice is consistent throughout. Our solution to this is to use *style files* which are locally defined mappings between names and directives in the markup language and parameters in the local system. For examples, speakers are referred to as “male1”, “female 2” etc in the markup language, but locally there is mapping saying which voices and parameters to use for each of these. In the case where only a single voice is available on the system, all speaker names in the marked up text will be mapped to this one voice. In a similar way, we have investigated the use of speaking style tags such as <RATE> which can be either internal to the document or not.

4. PRACTICAL USES OF STML

We have successfully incorporated STML into a number of applications requiring TTS output, including an email reader and a program which reads latex documents. The ILEX system is a natural language generation (NLG) system which examines a museum database and generates descriptions of exhibits taking into account the previous information that has been disclosed [6]. ILEX constructs its text from scratch and as such, has internal knowledge of the syntax and topic/focus structure of the text. It uses this structural knowledge to place STML tags in appropriate places in the text. If STML were not used the system could simply pass raw text, which would throw away all the information created in generating the sentence and force the TTS system to try to infer this (undoubtedly with mistakes) from just the words. Alternatively the two systems could be tightly coupled and the NLG system could write directly to the TTS system's internal data structures. This is often difficult from an engineering point of view as both the systems are large and an internal change in one often means changing some part of the other. STML is serves as a simple interface between the NLG and TTS systems, and doesn't limit the NLG system to using one synthesizer exclusively.

5. CONCLUSION

We have presented some preliminary specifications on STML, the Spoken Text Markup Language. As noted previously, we openly invite all interested TTS research groups to participate in future development of this standard. Further information on the current status of the project can be found at our websites, `www.cstr.ed.ac.uk/projects/stml.html` and `www.bell-labs.com/project/tts/stml.html`.

REFERENCES

[1] C. F. Goldfarb, *The SGML Handbook*. Oxford, Clarendon Press, 1990.

[2] C. Sperberg-McQueen and L. Burnard, *Guidelines for Electronic Text Encoding and Interchange*. Chicago/Oxford, Text Encoding Initiative, 1994. Available as http://www-tei.uic.edu/orgs/tei/info/elect.html.

[3] P. A. Taylor and A. Isard, "SSML: A speech synthesis markup language," *Speech Communication*, no. 21, pp. 123–133, 1997.

[4] Microsoft, *Microsoft Speech Software Development Kit Developer's Guide*, version 2.0 ed., 1996.

[5] T. V. Raman, *Audio System for Technical Readings*. PhD thesis, Cornell University, 1994.

[6] J. Hitzeman, C. Mellish, and J. Oberlander, "Generation of museum web pages: The intelligent labelling explorer," in *Proceedings of the Museums and the Web conference, Los Angeles, California*, 1997.