Integrating Language Generation with Speech Synthesis in a Concept to Speech System

Shimei Pan and **Kathleen R. McKeown**
Department of Computer Science
450 Computer Science Building
Columbia University
New York, N.Y. 10027
{pan, kathy} @cs.columbia.edu

Abstract

Concept To Speech (*CTS*) systems are closely related to two other types of systems: Natural Language Generation (*NLG*) and Speech Synthesis (*SS*). In this paper, we propose a new architecture for a *CTS* system. A *Speech Integrating Markup Language (SIML)* is designed as an general interface for integrating *NLG* and *SS*. We also present a *CTS* system for a multimedia presentation generation application. We discuss how to extend the current *CTS* system based on the new architecture. Currently, only limited semantic, syntactic and prosodic features are covered in our prototype system.

1 Introduction

Currently, there are two ways to develop a *Concept-To-Speech (CTS)* system. The first is to design a monolithic *CTS* system for a specific application. This design involves a specific *NLG* module and an *SS* module, often developed for the application, where discourse, semantic and syntactic information produced by the *NLG* module can be used directly by *CTS* algorithms to determine either system specific parameters for a *Text-To-Speech system*, or phonological parameters for a vocal tract model (e.g., (Young and Fallside, 1979)). One advantage of this design is its efficiency, but features from the two systems are usually so intertwined that the interface of the *CTS* algorithms are system dependent. Another design is to keep *NLG* and *SS* as independent as possible, thus allowing reuse of the current *NLG* tools and *TTS* systems for other applications. The typical design is equivalent to "*NLG* plus *Text-To-Speech*(*TTS*)" where the common interface between *NLG* and *TTS* is plain text. One advantage of this is its simplicity and adaptability. No change is necessary for existing *NLG* tools and *TTS* systems, but it suffers from a serious problem in that it loses useful information. All discourse, semantic and syntactic information is lost when the internal representation of *NLG* is converted to the text output and clearly this could be useful in determining prosody.

In this paper, we want to maintain the autonomy of *NLG* and *SS* so that they are reusable for different applications, yet flexible enough to easily integrate without losing useful information. We propose a new architecture in which the common interface is not plain text, but a *Speech Integrating Markup Language (SIML)*. We show how this architecture can be used in a multimedia presentation application where a prototype *SIML* was designed for this purpose.

2 Related Work

Recently, people have become more interested in developing *CTS* algorithms to improve the quality of synthesized speech. In (Prevost, 1995) and (Steedman, 1996), theme, rheme and contrast are used as important knowledge sources in determining accentual patterns. In (Davis and Hirschberg, 1988), given/new and topic structure are used to control intonational variation. Other *CTS* related research includes (Young and Fallside, 1979) and (Danlos et al., 1986). Most of the *CTS* systems developed to date have a closely integrated architecture. Because of this, *CTS* algorithms which map information from *NLG* to *TTS* parameters are system dependent.

There is some related research in developing markup languages for *TTS* and speech transcription. The *Speech Synthesis Markup Language*(*SSML*) (Isard, 1995) is used as an interface for *TTS*. The motivation behind *SSML* is to overcome the difficulty that different *TTS* systems require different input format. No additional information is provided as input to *TTS*, but *SSML* provides a straightforward representation of existing prosodic features. This

representation is too simple for the purpose of integrating *NLG* and *SS* for *CTS*. There is almost no discourse, semantic or syntactic information in their representation, yet these are features one would expect as output from *NLG* and which should influence the prosody of speech.

The *Text Encoding Initiative* (*TEI*) (Sperberg-McQueen and Burnard, 1993) provides a general guideline for transcribing spoken language using *Standard Generalized Markup Language (SGML)*. *SGML* is an international standard for encoding electronic document for data interchange. Integrating two components in *CTS* is a specific *SGML* application. Therefore, it can't be addressed directly in *SGML*. But the design of *SIML* can be guided by *TEI* standards.

3 System Architecture

The main new feature of the architecture (see Fig. 1) is the introduction of *SIML*. The system has three major components: the NLG component, the *SIML To Prosody* Component(*STP*) and the *TTS* component. Each can be designed and implemented independently. The *NLG-SIML* component first converts the input concepts into grammatical sentences with associated discourse, semantic, and syntactic information. Then the *SIML* converter transforms the system specific *NLG* representation into standard *SIML* format. The *STP* component computes the prosodic features based on the discourse, semantic and syntactic information encoded in the *SIML* format. The *STP* component has three modules: the *SIML* parser, the *STP* algorithms and the *SIML* generator. First the *SIML* parser analyzes the information in *SIML*. The *STP* algorithms predict prosodic parameters based on the information derived from the markup language. Then the *SIML* generator encodes the prosodic features in *SIML* format. The TTS component first extracts the prosodic parameters from the *SIML* representation and translates it into a specific, system dependent *TTS* input. In this way various *NLG* tools, *STP* algorithms and *TTS* can be integrated through the standard interfaces, *SIML*.

4 *MAGIC CTS* system

Our *CTS* system is a component of the *MAGIC* system (Multimedia Abstract Generation for Intensive Care). (Dalal et al., 1996) (Pan and McKeown, 1996). *MAGIC*'s goal is to provide a temporally coordinated multimedia presentation of data in an online medical database. The graphics and speech generators communicate through a media coordinator to produce a synchronized multimedia presentation. Given that *CTS* takes place within a multimedia context, many of the parameters our *CTS* system address are those needed for aiding coordination between media. Currently, there are three components in the *CTS* system: an *NLG* component, a set of *CTS* algorithms and a *TTS*. The *NLG* tools and *TTS* are application independent. We use the *FUF/SURGE* package (Elhadad, 1993) for generation. The speech synthesis system is AT&T Bell Labs' *TTS* system. The concept to speech algorithms, however, are not system independent. The input of these algorithms are in *FUF/SURGE* representation and the output is designed specificly for AT&T *TTS* input. In this section we describe our current *CTS* system and in the following section discuss extensions that we plan to adapt it to the general, proposed architecture.

NLG component in *MAGIC*

The *NLG* component in *MAGIC* consists of 4 modules: a general content planner, a micro planner, a lexical chooser and a surface realizer. The general content planner groups and organizes the data items from the medical database into topic segments; each segment may consist of several sentences. Then the micro planner plans the content of the sentences within each topic segment. One or several sentences can be used to convey the information within a topic segment. The lexical chooser makes decisions on word selection and the semantic structure of the sentence. The output of the lexical chooser is an internal semantic representation of the intended utterance.

For example, the internal semantic structure of "The patient is hypertensive" is represented in:

```
((cat clause)
 (proc
   ((type ascriptive)
    (mode attributive)))
 (partic
   ((carrier ((cat common)
              (head ((lex "patient")))))
    (attribute ((cat ap)
                (lex "hypertensive"))))))
```

In a semantic representation, a clause is defined by process type, participant and circumstance. Process type could be simple, as in the example, or composite (e.g., using conjunction). Each participant or circumstance may consist of a head and one or more pre-modifiers or qualifiers. Words and phrases are used to realize each semantic unit.

The surface realizer maps the lexicalized, semantic representation to its corresponding syntactic struc-

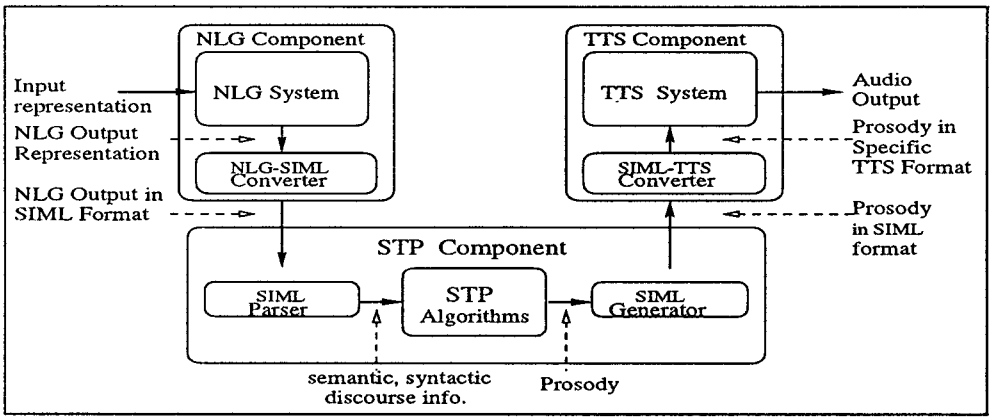

Figure 1: CTS System Architecture

ture. After linearizing the syntactic structure, which usually is the last step in a written language generation system, the internal semantic and syntactic structure as well as the words of the sentence are used as a rich and reliable knowledge source for speech synthesis.

CTS Algorithms in *MAGIC*

Due to the synchronization requirements, we are specifically interested in two features: pause and speaking rate. We want to increase or decrease the length of pauses or the speaking rate in such a way that speech actions begin and end at the same time as corresponding graphical actions. Even a small drift can be noticed by human eyes and cause uncomfortable visual effects. In *MAGIC*, only pause and speaking rate are set by our *CTS* algorithms; all other prosodic features are set by the default values predicted by AT&T Bell Labs' *TTS* system.

Currently, we use a simple strategy in adjusting the speech rate. We define the relative speaking rate as the ratio of the real speaking rate to the default speaking rate. Through experiments, we determined that the relative speaking rate can vary from 0.5 to 1 without significantly affecting the speech quality. In the future, we plan to develop an algorithm where the adjustable range is not uniform everywhere but decided by the underlying discourse, semantic and syntactic structures.

In the following, we give more detail on the *CTS* algorithm which is used to predict prosodic phrase boundary. It provides a reliable indication on where pauses can be inserted and how long the pause could be.

We use semantic structures to derive the prosodic phrase boundaries. In our algorithm, we first identify the basic semantic unit (*BSU*), which is the smallest, complete information unit in the semantic structure. Then we define the closeness measurement between two adjacent *BSU*s. If two adjacent *BSU*s are loosely connected, then we have reason to believe that it won't hurt the intelligibility significantly if we speak them separately. Therefore, semantic closeness is an important knowledge source for prosodic phrase boundary prediction. Other factors which also affect the placement of prosodic phrase boundary are breath length, and the distance to the end of the utterance.

A Basic Semantic Unit(*BSU*) is a leaf node in a semantic hierarchy. In the semantic hierarchy (see Fig. 2), the *BSU* is indicated by dark blocks.

We define the closeness between two adjacent *BSU*s as the level of the lowest common ancestor in the semantic hierarchy. If a node has only one child, then both parent and the child are considered at the same level. The closeness indicates the semantic distance of two adjacent *BSU*s. 1 means they are semanticly far apart, while higher numbers indicate they are semantically close.

Breath length is defined as the typical number of words a human can speak comfortably without breathing. The value used in the algorithm is learned automatically from a corpus. The distance from the current place to the end of an utterance is simply defined by the number of words.

Now we have 3 factors working together determining the prosodic phrase boundary. Basically, there won't be any prosodic phrase boundary within a *BSU*. For each place between two adjacent *BSU*s, we measure the possibility of inserting a prosodic phrase boundary using the combination of the 3 factors:

1. The larger the closeness measurement, the less the possibility of a boundary.

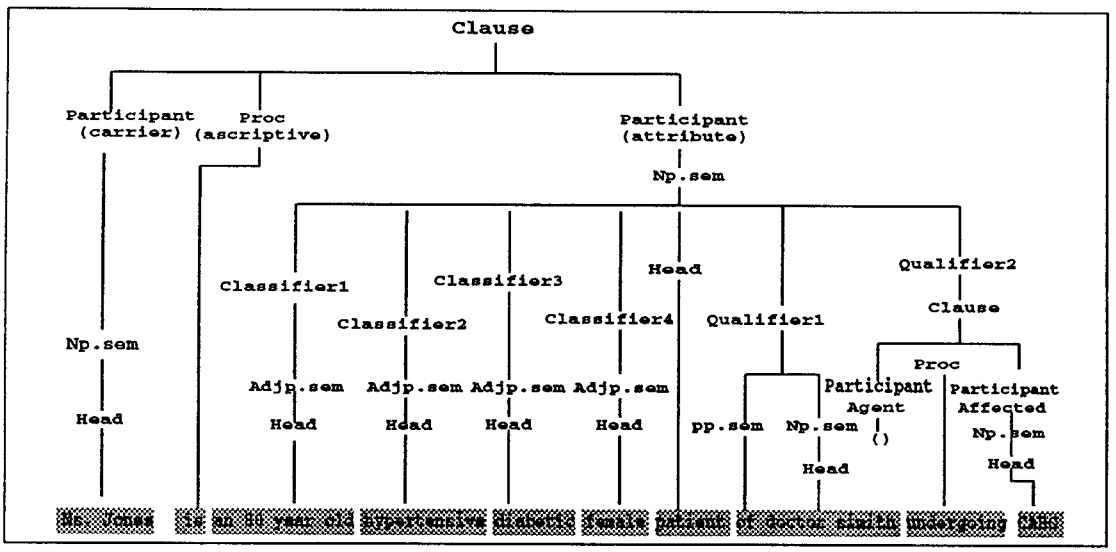

Figure 2: Semantic Structure and *BSU*

2. The closer the current breath length to the comfortable breath length, the more the possibility of a boundary.

3. The closer the current place to the end of the utterance, the less the possibility of a boundary.

4. The above factors are weighted, using a learning algorithm we trained automatically on a small corpus (40 sentences).

The result is encouraging. When we test this on the set provided in (Bachenko and Fitzpatrick, 1990), we got a 90% accuracy for primary phrase boundary and we get an 82% accuracy for the utterances in (Gee and Grosjean, 1983). We did not formally measure the algorithm for secondary phrase boundaries, because we only consider inserting pauses at primary phrase boundary.

TTS in *MAGIC*

Basically, we treat *TTS* as a black box in *MAGIC*. We use the escape sequence of *TTS* to override the *TTS* default value.

5 Extensions to *MAGIC CTS* Based on the New Architecture

The current *MAGIC CTS* uses *CTS* algorithms that are closely integrated with both the *NLG* tools and *TTS*. This will make it difficult to experiment with new tools, requiring changes in all the input and output format for the *CTS* algorithms. In the spirit of developing a portable language generation system such as *FUF/SURGE*, we are working on a portable spoken language generation system by using the new architecture.

Extension 1: Design *SIML* for *MAGIC*

In order to extend the current *CTS*, we must define a prototype *SIML*. As a first step, we have designed a prototype *SIML* that covers the information needed for *CTS* in the multimedia context. For our *CTS* algorithms, only semantic and syntactic structure are used in predicting prosodic phrase boundary and are represented in the *SIML*. Speaking rate and pause are also included in *SIML*.

We first describe how this information is represented in *SIML*, giving examples showing how to use *SIML* to tag pauses, speaking rate, semantic and syntactic structure. Then part of the formal Document Type Definition (*DTD*) of the prototype *SIML* is presented, providing a grammar for *SIML*. See (Sperberg-McQueen and Burnard, 1993) for more information about *SGML* and *DTD*.

Example 1: Using *SIML* to tag speaking rate and pauses:

```
<u.pro>Ms. Jones <pause dur=5 durunit=ms> is
an <phrase rate=0.9>  80 year old  </phrase>
hypertensive, diabetic female patient of
doctor Smith undergoing CABG. </u.pro>
```

<u.pro> and </u.pro> above indicate the start and end of an utterance. <phrase> and </phrase> is the front and end tag of a phrase. Rate is an attribute associated with <phrase>, indicating the speaking rate of the phrase. <pause> is a tag with

two associated attributes: dur and durunit. They indicate the length of the pause.

Example 2: using *SIML* to tag semantic structure:

```
<clause>
<participant role=carrier>
  <np.sem>
   The
     <head>patient</head>
  </np.sem>
</participant>
<proc type=ascriptive lex=be> is </proc>
<participant role=attribute>
  <adjp.sem>
    <head>hypertensive</head>
  </adjp.sem>
</participant>
</clause>
```

Example 3: using *SIML* to tag syntactic structure:

```
<sentence>
  <np> <art> The <noun> patient</np>
  <vp> <verb>is
       <adjp> hypertensive.</adjp>
  </vp>
</sentence>
```

Part of the formal definition of *SIML*, using *DTD*.

```
<!-- DTD specifying speaking rate and pause  -->
<! DOCTYPE utterance.pro [
<! ELEMENT u.pro - - ((#PCDATA| phrase|pause)*)>
<! ATTLIST u.pro
           rate  NUMBER 1 >
<! ELEMENT phrase - - ((#PCDATA| pause)*) >
<! ATTLIST phrase
           rate  NUMBER 1 >
<! ELEMENT pause - o  #EMPTY>
<! ATTLIST pause
          dur          NUMBER      #CURRENT
          durunit      CDATA       ms   >
]>
```

In the above DTD specification, three elements and their associated attributes are defined:

- u.pro and its attribute, rate;
- phrase and its attribute, rate;
- pause and its attributes, dur and durunit.

The following is the element definition for "u.pro":

```
<! ELEMENT u.pro - - ((#PCDATA| phrase|pause)*)>
```

ELEMENT is a reserved word for the element definition. "u.pro" is the element name. "- -" is an omitted tag minimization which means both the start and end tags are mandatory. The last part is the content model specification. (#PCDATA | phrase | pause)* means only parsed character data, phrases and pauses may appear between the start and end tags of "u.pro".

The associated attributes are defined in

```
<! ATTLIST u.pro
           rate  NUMBER 1 >
```

where the ATTLIST is the reserved word for attribute list definition. "u.pro" is the element name, "rate" is the attribute name, the type of "rate" is NUMBER and the default value is "1".

Extension 2: Design the *STP* component

The *STP* component is the core part in the architecture and deserves more explanation. There are three tasks for this component: parsing of the input *SIML*, generation of prosodic parameters from the information produced by *NLG*, and transformation of the parameters into the *SIML* format. The *SIML* parsing is straight forward. It can be done either by developing an *SIML* specific parser for better efficiency or by using an *SGML* parser (there are several which are publicly available). The output of this component is the semantic and syntactic information extracted from *SIML*. Generation of prosodic parameters must be done using a set of *CTS* algorithms; we need to change the input and output of our existing *CTS* algorithms and make it system independent. Since the performance of these algorithms directly affects the quality of the synthesized speech, much effort is required to develop good *CTS* algorithms. The good news is that the proposed design ensures that the markup to prosody algorithms are system independent. Therefore, they can be reused in other applications. The output of the *STP* algorithms then converts to the *SIML* format by the *SIML* generator. The procedure is straight forward and it can be done very efficiently.

6 Generalize *SIML*

Since the current prototype *SIML* is designed specifically for multimedia application, it includes very limited semantic, syntactic and prosodic information. Thus, it is currently too primitive to be used as a standard interface for other *CTS* applications. For the future, we must include other forms information that are needed for speech synthesis and that can be generated by an *NLG* system. Some types of knowledge that we have identified include:

1. Discourse information (e.g. discourse structure, focus, rhetoric relations etc.), semantic structure and its associated features (such as in the prototype *SIML*), and syntactic structure.
2. Pragmatic information such as speaker-hearer goals, hearer background, hearer type, speaker

type, emotions.

3. Morphology information, such as root, prefix, suffix.

4. Speech features, such as pronunciation, prosodic features, temporal information (such as duration, start, end), and non-lexical features (such as click, cough).

7 Conclusion and Future work

In this paper, a new *CTS* architecture is presented. The key idea is to integrate current *NLG* and *TTS* systems in a standard way so that the *CTS* system developed is able to use any existing *NLG* tools, *STP* algorithms and *TTS* systems and benefit from the information available from *NLG*. A Speech Integrating Markup Language is designed for this purpose.

In the future, we will extend our *STP* algorithms, to predict an adjustable range of speaking rate and stress placement based on discourse, semantic and syntactic information. As a result, we need to extend our *SIML* so that new information can be incorporated easily.

8 Acknowledgements

MAGIC is a system involving a large number of people at Columbia University. In addition to the authors, who are responsible for the text and spoken language generator, the *MAGIC* team includes James Shaw (media-independent content planning and text organization); Steve Feiner, Michelle Zhou (graphics generation); Mukesh Dalal, Li Yang (knowledge representation); and Tobias Hollerer (media coordination). We thank Becky Passonneau for providing the *SGML* developing environment. This research is supported in part by DARPA Contract DAAL01-94-K-0119, the Columbia University Center for Advanced Technology in High Performance Computing and Communications in Healthcare (funded by the New York State Science and Technology Foundation) and GER-90-2406.

References

J. Bachenko and E. Fitzpatrick. 1990. A computational grammar of discourse-neutral prosodic phrasing in English. *Computational Linguistics*, 16(3):155–170.

Mukesh Dalal, Steven Feiner, Kathleen McKeown, Shimei Pan, Michelle Zhou, Tobias Hollerer, James Shaw, Yong Feng, and Jeanne Fromer. 1996. Negotiation for automated generation of temporal multimedia presentations. In *Proceedings of ACM Multimedia 1996.*

L. Danlos, E. LaPort, and F. Emerard. 1986. Synthesis of spoken messages from semantic representations. In *Proceedings of the 11th International Conference on Computational Linguistics*, pages 599–604.

J. Davis and J. Hirschberg. 1988. Assigning intonational features in synthesized spoken discourse. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pages 187–193, Buffalo, New York.

M. Elhadad. 1993. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation.* Ph.D. thesis, Columbia University.

J. P. Gee and F. Grosjean. 1983. Performance structure: A psycholinguistic and linguistic appraisal. *Cognitive Psychology*, 15:411–458.

A. Isard. 1995. SSML: A markup language for speech synthesis. Master's thesis, University of Edinburgh.

Shimei Pan and Kathleen McKeown. 1996. Spoken language generation in a multimedia system. In *Proceedings of ICSLP*, volume 1, Philadelphia.

S. Prevost. 1995. *A Semantics of Contrast and Information Structure for Specifying Intonation in Spoken Language Generation.* Ph.D. thesis, University of Pennsylvania.

C.M. Sperberg-McQueen and L. Burnard. 1993. *Guidelines for Electronic Text Encoding and Interchange.* ACH, ACL and ALLC.

M. Steedman. 1996. Representing discourse information for spoken dialogue generation. In *Proceedings of the International Symposium on Spoken Dialogue*, pages 89–92, Philadelphia.

S. Young and F. Fallside. 1979. Speech synthesis from concept: a method for speech output from information systems. *Journal of the Acoustical Society of America*, 66:685–695.