SOCS: A SPEECH OUTPUT SYSTEM FROM CONCEPT REPRESENTATION

Yoichi Yamashita and Riichiro Mizoguchi

The Institute of Scientific and Industrial Research, Osaka University,
8-1, Mihogaoka, Ibaraki-city, Osaka, 567 Japan.
yama@ei.sanken.osaka-u.ac.jp

ABSTRACT

This paper describes a Concept-To-Speech (CTS) conversion system, SOCS (Speech Output from Case Structure representation). For CTS systems, how to represent the input to the system is a very important issue to discuss. In SOCS, a representation based on case structure and phrase patterns, which is called concept representation, is converted into synthetic speech. SOCS plays a crucial role in a speech output interface which is designed as a general frontend of various intelligent performance systems (IPS). The message transmission of the concept representation satisfies the requirements both from IPS and the speech synthesis component. SOCS has two new built-in mechanisms for direct control of prosodic parameters by a concept representation. The first one is the pause marker which is generated along with words in the sentence generation. The pause marker is used both to insert pauses and to set the boundaries of the prosodic phrase component of f0. The second one is the Prosody Modification Functions (PMF) embedded in the custom templates. PMF controls the the prosodic parameters for the prepared sentence pattern. The dialog management in a speech output interface based on SOCS is also discussed.

1. INTRODUCTION

Most of the current Intelligent Performance Systems (IPS) generate messages in text which will be output on visual display terminals. However, there is no reason why the messages generated by IPS should be represented in text when we exchange information with IPS through speech. In speech synthesis, the problem of "from what speech is converted" is a crucial issue to discuss. Text which is written in natural language is usually the source of synthetic speech, and this speech synthesis technique is called Text-To-Speech (TTS). For human-computer interaction through speech, however, other representation forms can be more convenient both for the speech synthesis component and IPS rather than text. Young and Fallside have developed a Concept-To-Speech (CTS) conversion system, SSC (Speech Synthesis from Concept), which converts the input concept from an information system into speech using word concatenation synthesis[1]. In general, the input to CTS is described with a semantic representation scheme rather than text.

To realize very natural dialog between human and computer, there are many problems to be solved. Half of them are concerned with general dialog processing, such as the complement or the ellipsis of words in utterances, the understanding of human user's intention, the generation of the most suitable expression for the situation, and so on. The other half are concerned with human-computer interaction through speech, such as the recovery from speech recognition errors, the generation of prosody of the output speech, and so on. Such problems are not dependent on the process in IPS, but also inherent in dialog. Therefore, a flexible speech interface component which is separated from IPS should deal with these dialog phenomena. The intermediate representation in terms of natural language sentence makes it difficult to construct the flexible speech output interface, because the expression of output speech suitable for the dialog context should be modified during the dialog process dynamically, and the modification of the natural language sentences in the interface is very difficult. The above observation shows the messages from IPS to the interface should be described with a more flexible intermediate representation scheme rather than the natural language sentence.

CTS technique plays an important role for speech output using the framework of the general speech interface mentioned above, since it has several advantages over conventional TTS conversion systems; (1) there is no need for text analysis because CTS generates sentences from the concept representation itself, (2) some prosodic features, such as emphasis, utterance speed, and so on, can be embedded in the concept representation, and (3) a concept representation can be transformed into various sentences according to dialog context. IPS should determine only what to say not considering how to say. Thus, CTS is one of necessary techniques for human-computer interaction through speech.

2. SPEECH OUTPUT INTERFACE AND SOCS

Summarizing the above discussion, we obtain the following three requirements for designing speech output interface.

(1) The mechanisms of the interface should not be dependent on a particular IPS, and data and knowledge prepared for the IPS should be separated from other general components of interface.
(2) The intermediate representation form which describes messages from IPS to the interface should be flexible enough for various modification.
(3) The components of the interface, such as the dialog manager, the sentence generator, and the speech synthesizer, should be tightly coupled with and exchange information with one another.

Fig.1 shows a block diagram of the speech interface including speech input. SOCS (Speech Output from Case Structure representation) is the proposed CTS system which converts the concept representation into speech. The intermediate representation form generated by IPS is called the concept representation and is sent to SOCS via the dialog manager. In SOCS, IPS-independent procedures for sentence and prosody generation are realized by introducing a flexible concept representation scheme, which is based on the case structure and several phrase patterns. The dialog manager modifies the concept representations according to the dialog context before the sentence generation gives some information to prosody decision process in order to synthesize natural speech.

3. CONCEPT REPRESENTATION

3.1. Requirements for Intermediate Representation Form

A lot of representation schema, such as natural language sentence, phonetic transcription, semantic network, syntactic representation, and so on, could express messages from IPS to

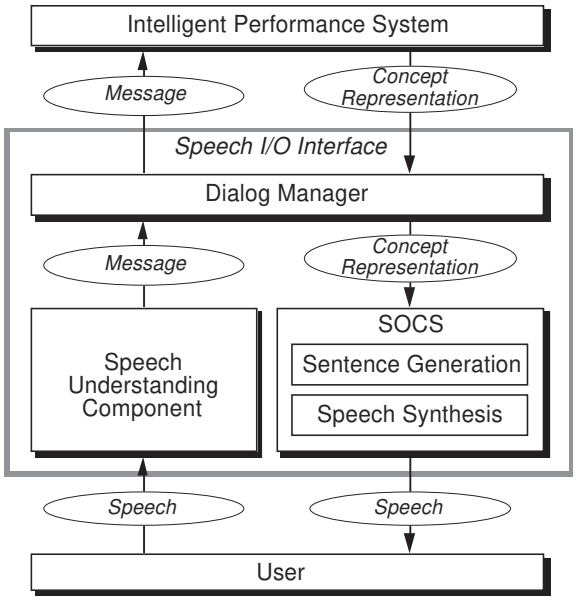

Fig.1 A block diagram of speech interface.

CTS. For a general CTS system serving as a frontend of various IPSs, the most important issue to discuss is what intermediate form represents the input to CTS. Generally speaking, the representation scheme for message description plays a crucial role in our problem.

For the determination of the intermediate form which represents the input to CTS, there are six criteria from the viewpoint of CTS and IPS as follows:

C1: It must be easy to extract the information necessary for prosody control from it. One of the main reasons why representation schema other than text are investigated is that it is expected to make it easier to control the prosodic parameters, which shows this criterion is essential to our system.

C2: It must be easy to generate sentences. CTS must generate sentences from the intermediate representation form, while TTS must analyze the input text. Therefore, it is desirable to accomplish the sentence generation with a simple mechanism.

C3: It must be easy to modify the intermediate form for generating sentences appropriate for the dialog context. IPS only has to make concept representation in terms of the intermediate forms, not taking account of the dialog context. This criteria is needed because a dialog manager, which locates between IPS and CTS, must adapt the intermediate forms to the dialog context by doing some modification, such as deletion of words, addition of prosodic features, and so on.

C4: It must be easy to prepare the data which is necessary for the intermediate form generation module in IPS when IPS is implemented.

C5: The intermediate form must be expressive so that one representation can be converted into several speech expressions of different sentences and prosodic patterns. It is not desirable that a lot of data must be prepared for the output generation module of IPS.

C6: It must be easy for IPS to make the intermediate forms. IPS converts the result of the inference or the information processed in it into the intermediate form during the course of execution. This criterion guarantees that this conversion is easy.

First three criteria are requirements from CTS, and the others are from IPS. There is a trade-off in determination of the representation scheme, since requirements from IPS and CTS contradict each other. The use of a very abstract representation scheme imposes a tough task of sentence generation on CTS. On the other hand, a tightly syntax-specified representation such as natural language sentence decreases both the flexibility in its modification and the variation of output messages. We compared six sorts of representation schema as the input to CTS: natural language sentence, phonetic transcription, template-based representation, syntactic tree, case structure-based representation, and CD-representation[2], and it was found out that the combination of case structure-based representation and some templates satisfies six requirements mentioned above and is the most appropriate representation scheme for the input to CTS. Therefore, the SOCS system employs a sophisticated intermediate form based on case structure and several phrase patterns which are provided as custom templates. Fig.2 shows three examples of intermediate forms, which are called concept representations. In this representation scheme, the meaning of the message and most words used in the surface sentence are given, but the syntactic structure of the surface sentence is not determined except for the use of the phrase pattern. The characteristic of the concept representation removes the complexity of sentence generation process in CTS, and introduces the variety of the generated sentences according to the dialog context.

```
s1( hajimeru([$object(lesson1),$persuasive]) ).
      ('start')
  lesson1( $modify( chiri, benkyo,  []) ).
                ('geography') ('lesson')
s2( $because(soviet1,seiikusuru([$actor(kome ),$negative])) ).
                              ('grow')          ('rice')
   soviet1( $is( soren, $modify( samui, kuni, [$p_prominent(samui )]) ).
          ('the Soviet union')     ('cold') ('country')          ('cold')
s3( $why(seiikusuru([$actor(kome),$place(iran)]),[$p_interrogative]) ).
               ('grow')          ('rice')        ('Iran')
```

Fig.2 Examples of concept representations.

3.2. Constituents of Concept Representation

Our concept representation is defined based on the case structure and is composed of the seven constituents: atomic symbol, modification template, case template, mood operator, conjunctive template, custom template, and prosody operator. In Fig.2, every string beginning with the '$' symbol in the concept representations is a template or operator defined in the SOCS system. For example, the '$why()' of s3 is a custom template.

Words such as verbs, nouns, and adjectives appear as atomic symbols in a concept representation. The modification template, '$modify(A,B,OPs)', represents a modification relation between the two arguments, 'A' and 'B', which are constructs of sentences, such as nouns, adjectives, and phrases. The case templates are prepared to represent the case information of verbs and are used as the arguments of verbs in the concept representation. The case information describes the relation between actions and objects, and 13 case templates are defined in SOCS. The mood operators describe the tense, aspect, and modality information of verbs. The mood operator changes the verb into an appropriate form according to the prestored pattern. The conjunctive templates represent the relation between the two consecutive sentences. The custom templates express some sentence/phrase patterns which are frequently used in the output messages of IPS and those whose semantic structure are not based on the case relations. Finally, the prosody operators are associated with verbs and templates to indicate the prosodic features. SOCS has three prosody operators ('$p_prominent(A)', '$p_interrogative', and '$p_speed(A,S)').

```
$reason( A, B, OPs ) :  [ A, wa, p2, B, kara, desu ],  $mod_acc(4,+,15), $mod_bpit(4,+,10).
                        ( 'the reason of A is B' )
$because( A, B, OPs ) :  [ A, kara, p2, B ],  $mod_bpit(4,190).
                         ( 'Because A, B' )
$why( A, OPs ) :  [ naze, p2, A, no, desu, ka ],  $mod_dur(1,+,10), $mod_bpit(3,190).
                  ( 'why A' )
$is( A, B, OPs ) : [ A, wa, p1, B, desu ].
                   ( 'A is B' )
```

Fig.3 Examples of custom templates.

s1: *chiri no* p1 *benkyo wo* p1 *hajimemashou*
('Let's start studying geography.')

s2: *soren wa* p1 *samui kuni desu kara* p2 *kome ga* p1 *seiikushimasen*
('As the Soviet union is a cold country, rice does not grow.')

s3: *naze* p2 *kome ga* p1 *iran de* p1 *seiikusuru no desu ka*
('Why does rice grow in Iran?')

Fig.4 Examples of generated sentences.

Since the custom templates plays an important role in SOCS to make prosodic patterns, details of them are discussed here. The examples of the custom templates are shown in Fig.3. A custom template generates a sentence or a phrase using the output word list, which is denoted in brackets in Fig.3. This list can contain pause markers, 'p1' and 'p2', as well as words. Pause markers are embedded in sentences generated and are utilized during the prosody control process discussed in section 5.1. Pause markers are also used in other templates: the modification templates, the case templates, and the conjunctive templates. Furthermore, Prosody Modification Functions (PMF) can be associated with the custom template, and locally control the prosodic parameters, such as duration, f0 at the onset of the prosodic phrase, and the accent prominence of words. The codes '$mod_acc' and '$mod_bpit' in the '$reason' template are examples of PMF. PMFs also facilitate the control of prosody for these expressions.

4. SENTENCE GENERATION

Sentence generation is carried out according to the template invocation and the case grammar. A sentence generated includes pause markers and syntactic structure. Verb phrases are generated by concatenating their case elements in a pre-determined order according to case descriptions. In sentence generation, the sentence is automatically changed into Japanese polite expression, which ends sentences with function words *-desu* or *-masu*, so that SOCS becomes a friendly interface for the user of IPS.

Fig.4 shows the sentences generated from the concept representations in Fig.2. A verb *hajimeru*, which appears in s1 of Fig.2, is first changed into its polite expression *hajimemasu*. Then, the '$persuasive' operator adds function words *u* after *masu* is inflected into *masho*. The sentence generation of *hajimeru* of s1 results in *hajimemashou*. The template has the list of output words which may include pause markers, 'p1' and 'p2'. The sentence generation is carried out with pause markers embedded in sentences as shown in Fig.4.

5. PROSODY CONTROL

5.1. Pause Marker

Pauses are inserted according to the pause markers embedded in the generated sentences. The marker 'p2' dictates the insertion of a long pause (300ms), PS2, and the marker 'p1' indicates the potential insertion of a short pause (100ms), PS1. Some of the 'p1's are replaced by PS1, taking account of number of morae between the pauses, while every 'p2' is always replaced by PS2. If the number of morae in a one-breath phrase is over 25, one of 'p1's in the phrase is replaced by PS1 to divide the phrase into two one-breath phrases. Note that words and pause markers are generated in a nested structure. A 'p1' from the outermost template is selected, when more than one 'p1' are found in the phrase to be divided. This replacement of pause markers is recursively carried out from the outside in, until the number of morae in every one-breath phrase becomes less than the threshold (=25). The boundary between prosodic phrases is also determined in the same way as the pause insertion, except for the use of a smaller mora threshold (=15).

5.2. Prosody Modification Function (PMF)

The Prosody Modification Functions (PMF) are used only in the custom templates to locally change the prosodic parameters in the sentences. In the current implementation, three PMFs, '$mod_dur', '$mod_bpit', and '$mod_acc', are prepared to control the duration, the f0 frequency at the onset point of prosodic phrase, and the accent prominence component, respectively. A PMF with two arguments absolutely sets the value of the parameter and one with three arguments does so relative to the default value. In both, the first argument indicates the position of the elements in the output word list which is operated by PMF. The third argument of '$mod_dur' is interpreted as a percent ratio of the default value.

5.3. Prosody Operators

The prosody operators are given only in the concept representation, while PMFs are used only in the custom templates. In other words, PMFs are associated with the sentence pattern and the prosody operators are used by IPS and a dialog manager to partially change prosodic parameters of the output speech according to the context of the dialog. Three prosody operators are currently defined: '$p_prominent(A)', '$p_interrogative', and '$p_speed(A,S)'. The operator '$p_prominent(A)' increases the accent prominence by 50Hz, partially lifts the declination line by 40Hz, and lengthen the duration by 5% for its argument 'A', in order to prosodically emphasize it. The operator '$p_interrogative' requires that f0 rise at the end of the sentence. SOCS prepares this special f0 pattern for the interrogative sentence. The operator '$p_speed(A,S)' controls the utterance speed of its first argument 'A' according to the second argument 'S'.

5.4. An Example of Prosodic Parameter generation

Fig.4 shows a result of prosodic parameter generation for s3 of Fig.2. The pause marker 'p2' from the custom template '$why' inserts a long pause between the two phrases. The '$mod_bpit' PMF associated with the '$why' template sets the f0 frequency to 190Hz at the onset point of the second prosodic phrase beginning with *kome* ('rice'). In s3, one prosody operator, '$p_prominent', is used in the concept representation to emphasize the word *samui* ('cold') and the '$p_prominent' increases the f0 frequencies for *samui*.

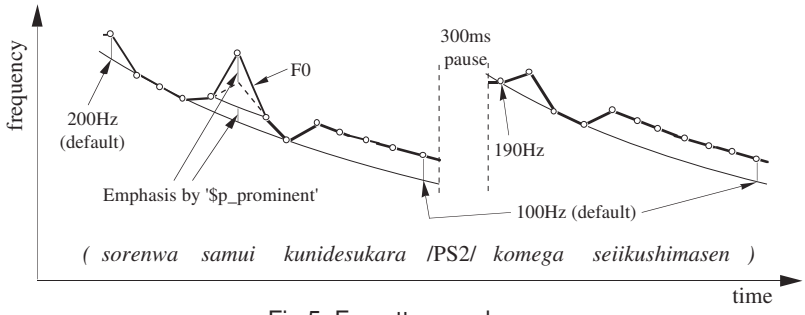

Fig.5 Fo pattern and pause.

6. DIALOG MANAGEMENT IN SPEECH OUTPUT INTERFACE

In the speech output interface incorporating the SOCS system, shown in Fig.1, the dialog manager can modify concept representations generated by IPS so that the output speech matches with the dialog context and it is natural especially in prosody. Two kinds of dialog processing are described in this section.

6.1. Extraction of Emphatic Words

How to prosodically emphasize words and phrases in utterances of IPS is classified into three categories: (1) partial replacement and repetition of words appearing in a surface sentence, (2) the major point of IPS's answer to the user's question, and (3) the use of the domain knowledge of IPS. Last category is not dealt with in the dialog manager because only IPS knows what should be emphasized. For example, in an utterance of IPS which guides a tourist, "please take the subway of Ginza line at Shinjuku," "Ginza" should be emphasized because there is another line at Shinjuku. However, the dialog manager does not know that fact. For the framework of the general speech output interface, IPS must indicate such emphases in a concept representation. The dialog manager is responsible for the first two categories of emphases.

The first category of words to be emphasized is easily extracted by the dialog manager based on the comparison of concept representations between IPS's and the preceding user's. For example, when IPS says "please take the subway at Shinjuku" after the user asked "I take the subway at Ikebukuro?", "Shinjuku" is extracted as an emphatic word because it replaces a word, "Ikebukuro", in the user's utterance. Of course, this is carried out on the level of the concept representation.

The second category of emphatic words must be extracted by identifying the intention of user's question, that is, what the user wants. We have proposed the dialog management system for speech understanding using SR-plans and twin stacks[3]. After the user's intention is identified by using such a technique, the dialog manager can extract the major point of IPS's answer based on it using concept representation. For example, suppose that IPS generates the output message of "please get off the subway at Tokyo" using concept representation:

oriru([$object(*chikatetsu*),$place(*tokyo*),$imperative])
'get off' 'subway'

after the user asked to IPS the station where he/she should get off the subway. As the dialog manager knows that the major point of answer is information about a place if the intention analysis is successfully done, the dialog manager changes the concept representation into

oriru([$object(*chikatetsu*),$place($p_prominent(*tokyo*)),
$imperative])

to prosodically emphasize '*tokyo*' in SOCS. Since the concept representation describes that '*tokyo*' is information about a place, the dialog manager finds out the emphatic word with such facility even though it does not know '*tokyo*' is a name of the place.

6.2. Use of the Custom Templates

In SOCS, the custom templates are prepared for the frequently used expression. As they describe the prosodic pattern using PMFs, the use of them in the concept representations can synthesize more natural speech. IPS does not necessarily know what custom templates it can use. The dialog manager rewrites the concept representation generated by IPS in terms of custom templates, if it is possible. For example, a concept representation,

hiraku([$object(*kaigi*),$place(*doko*),$interrogative])
'hold' 'conference' 'where'

can be rewritten into

$where(*hiraku*([$object(*kaigi*)]).

7. CONCLUSIONS

A new idea for a Concept-To-Speech (CTS) conversion system has been proposed. The input of the Speech Output from Case Structure representation (SOCS) system is case structure-based concept representation which is described at an abstraction level appropriate both for IPSs and CTS. The use of the concept representation rather than natural language sentence makes it possible to modify the output sentences according to the dialog context and to construct the flexible speech output interface. The SOCS system directly controls prosodic parameters using the concept representation with two built-in mechanisms: the Prosody Modification Function (PMF) in the custom template and the pause marker. In SOCS, the custom templates play an important role and generate the prosodic parameters for the prepared patterns. Since many IPS systems have output message patterns which are frequently used to answer the user's questions and to query, it is very effective for the CTS system to prepare such patterns in order to generate very natural synthetic speech. A prototype SOCS system has been implemented in Prolog a UNIX engineering workstation and connected to an intelligent tutoring system developed at our laboratory.

Reference

[1] Young,S.J., and Fallside,F.: "Speech synthesis from concept: A method for speech output from information systems," J.A.S.A., 66, 3, 685-695 (1979).

[2] Schank,R.C.: "Conceptual Information Processing," Vol.3 of Fundamental Studies in Computer Science, North-Holland (1975).

[3] Yamamoto,T., Ohta,Y., Yamashita,Y., Kakusho,O. and Mizoguchi,R.: "MASCOTS: Dialog Management System for Speech Understanding System," Trans. IEICE, Vol.E74, 7, 1881-1888 (1991).