PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 33, NO. 4, PP. 183-193 (1989)

# TEXT-TO-SPEECH SYNTHESIS FOR DUTCH: DEVELOPMENT AND INTEGRATION OF DIFFERENT KNOWLEDGE SOURCES

B. VAN COILE

Laboratory of Electronics and Metrology,
State University of Ghent, Belgium

Received June 15, 1988.

## Abstract

This paper deals with the development of a text-to-speech system for Dutch. Special attention is paid to a highly flexible development tool called DEPES. This tool provides both a powerful knowledge representation language which is very well suited for text-to-speech problems, and a compiler program which accepts (linguistic) grammars written in this language.

The DEPES software tool was used successfully to create and implement many knowledge sources in our text-to-speech system.

*Keywords:* Speech synthesis, text-to-speech system.

## Introduction

Today, the interest in speech synthesis and speech recognition techniques is still growing. Considering the fact that speech is man's primary and most natural means of communication, it is obvious that all over the world research institutes and firms are trying to use speech to communicate with computers. A man-machine interface which uses verbal communication requires a speech recognition and a speech synthesis system. This paper deals with the synthesis part and especially with the development of a text-to-speech synthesis system for Dutch.

Computer voice response can be achieved in different ways. If only a small number of fixed messages have to be spoken by the computer, it may be appropriate to record and store each message separately. It is obvious that this method does not require any knowledge about linguistic processes. Other systems use a vocabulary of human spoken words for message assembly. However, the simple concatenation of words into whole messages usually results in unnatural sounding speech. Several reasons can be indicated. Acoustic realisations of the same word in several contexts show clear differences near the word boundaries. These differences are due to phonological processes and coarticulation. Furthermore, the prosodic characteristics of a message (intonation, stress pattern and rhythm) are

highly disturbed if single words are simply concatenated. In order to solve these problems a lot of phonological, syntactic and phonetic knowledge is needed. The most important disadvantage of speech synthesis by means of the word concatenation technique is obvious: even with an enormous amount of words stored, the synthesizer will not be able to convert every message into speech. To solve this problem text-to-speech synthesizers use a finite set of elementary speech segments such as half-syllables, diphones or phonemes to convert the unrestricted text into speech. It is obvious that such a system needs a lot of knowledge (data and rules) to imitate the complex cognitive process of reading aloud.

## General outline of text-to-speech synthesis

Most text-to-speech systems consist of three main parts: a linguistic, a phonetic and a synthesizer part (*Fig. 1.*). In the linguistic part, the incoming unrestricted text is given a phonetic representation: this representation can be seen as a phonetic transcription of the text supplemented by information on for example syntax and stress. In order to create this representation, a lot of high level linguistic knowledge is needed. The use of special knowledge representation methods and AI-techniques is obvious if not indispensable.

The phonetic part of the system uses the information of the linguistic section to produce parameters for a speech synthesizer. It incorporates rule-based strategies and algorithmic routines for the creation of a correct prosodic pattern and for the synthesis of correct spectral characteristics (segmental synthesis).

In the third part of the system, the speech parameters are used by the synthesizer in order to obtain the acoustic waveform. It is obvious that this part performs an algorithmic and essentially numeric task.

If we examine the different steps of text-to-speech conversion, we find that a lot of different cooperating knowledge sources are necessary.

## The DEPES system

When linguistic knowledge is hard-coded into a computer program, it is difficult to make changes. Consequently the development and testing of new rule grammars is not so easy. We believe that the development of such grammars is aided and speeded up considerably by the use of a rule compiler which offers the possibility of expressing rules in an easy-to-read special rule language and to test rule grammars easily (CARLSON and GRANSTRÖM, 1976; HERTZ, 1982; HERTZ et al, 1985).

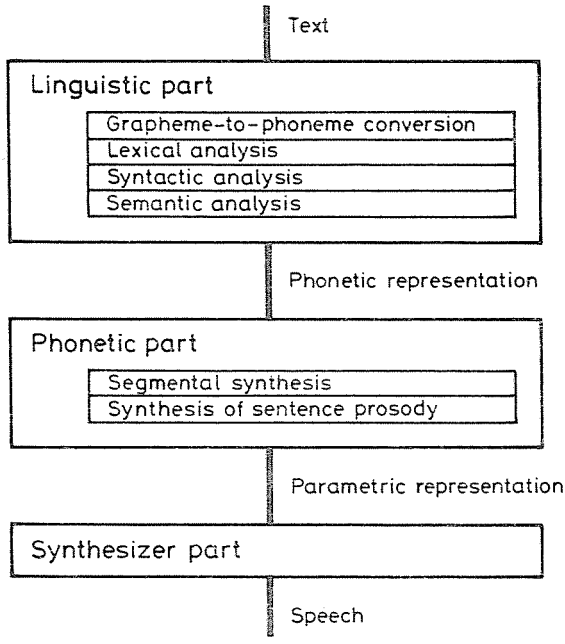


*Fig. 1.* General outline of a text-to-speech system

At the laboratory we implemented a powerful development tool named DEPES (Development Environment for Pronunciation Expert Systems). The different components of DEPES are shown in *Fig. 2.*

There are three executable programs (which are written in Pascal): the pre-compiler, the pre-linker and the dictionary service. The pre-compiler accepts a rule grammar written in the powerful DEPES knowledge representation language based on the formalism used in Generative Phonology. The pre-compiler converts the rule grammar into DEPES object code. The pre-linker is used to combine the object files of different grammars and to create a Pascal module and a common loadable data structure. The dictionary service accepts a dictionary data file and creates a loadable dictionary and also a Pascal module. The Pascal modules created by the DEPES development tool have to be compiled using the ordinary Pascal compiler. Then they can be linked with other DEPES modules and with non-DEPES modules (which are, for example, written in Pascal or C). In order to minimize the compilation and linking delays the DEPES-programs

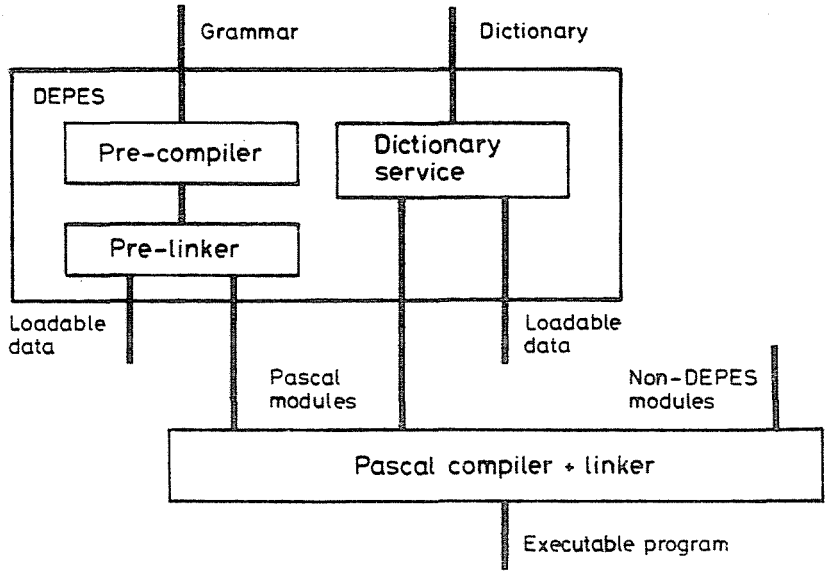


*Fig. 2.* The DEPES text-to-speech development system.

check for differences between successive versions of grammars or dictionaries. If only the loadable data structures are changed, no further processing is necessary and the program can be re-run with the new version of the dictionary or grammar.

The DEPES development tool relieves the user of as many details as possible and permits concentration on the linguistic problem. Even people without any programming experience (the majority of the linguists) can develop and test their own linguistic rule grammars.

DEPES is a dedicated system. Thanks to this dedicated approach it was possible to combine ease-of-use with efficiency as far as speed and storage are concerned. The development environment is available on PC. In order to create quasi real-time implementations of linguistic grammars on our 8086/TMS32010 biprocessor card, the development system also offers the possibility of choosing Pascal-86 (from Intel) as a destination language dialect. Consequently, new grammars or enhanced grammars can be incorporated very easily and rapidly into our text-to-speech synthesis program which runs on this biprocessor card.

## The DEPES rule language

The linguistic rules operate on a central data structure which consists of an array of records, each record contains one or more fields. In its simplest form the central data structure is an array of characters, i.e. a single string. *Fig. 3.* shows a simple data structure with only two fields called *GRAPHEMES* and *PHONEMES*. The *GRAPHEMES* field is initialised with the Dutch word 'bad' (bath).

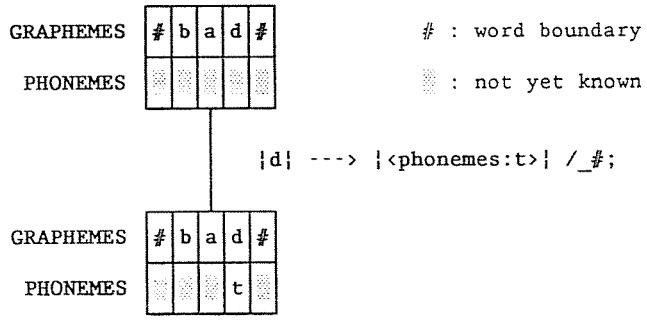


*Fig. 3.* The rule formalism used in DEPES

Speakers of Dutch know that this word has to be pronounced as /bAt/ (ASCII symbols replace IPA phonetic symbols). However, they are unaware of the fact that they are using the following linguistic rule for Dutch pronunciation: 'a letter [d] becomes a phoneme /t/ when the letter occurs at the end of a word.' Rules of this kind can be written using a formalism taken from Generative Phonology:

$$A \longrightarrow B \ /X\_Y$$

The pattern $A$ in the data structure (also called the focus of the rule) is changed into the pattern $B$ if the focus $A$ is preceded by a left context $X$ and followed by a right context $Y$. An ordered set of such linguistic rules is called a (linguistic) grammar.

*Fig. 4.* shows a small example of a file with a grammar. Such a grammar file has a classic program structure: a header, a declaration section and a body. The header includes the name of the grammar. The declaration section includes the definition of the central data structure, linguistic features, structured features, variables and external procedures.

—the *central data structure* consists of one or more fields. Each field has its own identifier and type (character or integer). Field identifiers are

```
Module Grapheme-to-phoneme;
(* SOME rules which are used in the
   transcription of Dutch words *)

Data_structure    phonemes : char;

Feature     schwa            = [$];
            plosive.voiced   = [bdG];
                   .unvoiced = [ptk];
            nasal            = [mnN];
            space            = [ ];
            cons             = plosive+nasal+
                               [fsSHvzZglrwjh];
            vowel            = [aAeEiIoOuy^0@YJ$];


Var     C1 : cons;


begin
¦o¦ ---> ¦O¦ /_[cons]{[cons],#};                    (1)
¦e¦ ---> ¦$¦ /[+vowel,-schwa][-space,(0,)]_n;       (2)
¦e¦ ---> ¦$¦ /{g,b}_[-space,(0,)][vowel,-schwa];    (3)
¦[C1][C1]¦ ---> ¦[C1]¦;                             (4)
¦plosive.voiced¦ ---> ¦plosive.unvoiced¦ /_#;       (5)
end.


Remark:  [-space,(0,)] :
          a sequence of non-blanks (zero or more)


Some derivations:
```

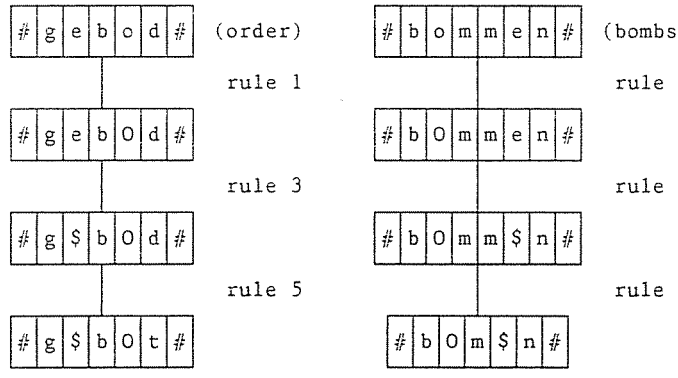


*Fig. 4.* Example of a linguistic grammar.

used inside rules in order to change between fields. This is exemplified by *Fig. 3.*: the rule performs the pattern checking on the default field *graphemes* while the modification is done on the *phonemes* field. The example grammar in *Fig. 4.* uses a very simple central data structure which consists of only one character field.

—a *feature* can be seen as a set of characters. Using such feature definitions, it is possible to combine many similar elementary rules into one easy-to-read super-rule. For example, the first rule in *Fig. 4.* states that the long vowel /o/ has to be changed into the short vowel /O/ if the /o/ is followed by two consonants or by one consonant and a word boundary.

—*structured features* are used to define sets of characters and to establish a relationship between these sets. The fifth rule of the example in *Fig. 4.* uses the structured feature *plosive* to change voiced plosives at the end of a word into the corresponding unvoiced ones.

—a *variable* is defined as an element of a feature set. Used in a rule, variables originally don't have a value. During the pattern matching process variables receive a value. Within one rule, variables are interpreted consistently, while variables in different rules are completely independent, even if they have the same name. Rule 4 of the example (*Fig. 4.*) states that when two *identical* consonants are combined, one has to be omitted.

—a linguistic grammar can call *external* grammars, dictionaries or Pascal routines. *Fig. 5.* shows how the grammar *sentence* uses two external facilities: the dictionary *exceptions* and the grammar *word.* For each maximal sequence of 1 or more non-blanks (i.e. one word) the dictionary is consulted. On failure, the grammar *word* is called.

```
grammar sentence;
data_structure      phonemes : char;
feature             space = [ ];
ext                 exceptions : dictionary;
                    word       : grammar;
begin
¦[-space,(max,1)]¦ ---> exceptions;
                    ---> word;
end.
```

*Fig. 5.* Using external grammars or dictionaries

The body of the grammar is block structured. Each block consists of an ordered list of linguistic rules. The header of a block describes the inference control mechanism applicable to the block. In its simplest form, the body of the grammar consists of one block without an inference control

directive. This indicates the use of the default inference mechanism which can be described as follows:

—The rules are used in order of appearance. This means that the rules are scanned once from top to bottom.

—For each rule (before going to the next rule) the central data structure is scanned from left to right. Whenever pattern matching is possible, the tested rule is applied and the data structure modified. All subsequent testing is done on the modified data structure.

The default inference mechanism was used for the derivations shown in *Fig. 4*. Several other inference mechanisms are also available. They are activated by the appropriate inference control attribute in the header of the block.

—The attribute *do until first* is used to force a block exit as soon as a rule is applied.

—The default scanning direction of the central data structure is *from left to right*. The block attribute *from right to left* makes it possible to use the opposite scanning direction.

—Using the block attributes *rule until first* or *rule until last*, it is possible to reverse the default nesting of the rule scanning and the scanning of the central data structure.

The DEPES development tool and rule language just described, were used successfully for the development and the integration of many knowledge sources in our text-to-speech synthesis system for Dutch. It has to be emphasized that the introduction to DEPES just given, is only a brief and incomplete description of the system. A lot of interesting features are beyond the scope of this paper.

## A text-to-speech system for Dutch

### *The linguistic section*

The linguistic section of our present synthesis system deals almost entirely with the conversion of the input text into its phonetic transcription. Only a rudimentary lexical and syntactic analysis is performed.

In order to create the phonetic transcription of the text, the system performs the following tasks:

—deal with special character sequences such as digit strings, telephone numbers, date and time indications, abbreviations, alphanumeric strings, special symbols etc.

—realize the syllabification of the words

—make a grapheme-to-phoneme conversion
—assign lexical stress.

We have developed several linguistic grammars applicable on different items such as single words, whole sentences, digit strings etc. The total number of linguistic rules is about 500. The most complex and most important grammars deal with the orthographic-phonetic transcription of single words. The necessary linguistic knowledge was obtained using an orthographic-phonemic dictionary of the 10,000 most frequent words of Dutch. The present system uses almost 300 rules to cover 91% of these 10,000 words (96% if we omit lexical stress, VAN COILE, 1987a). Those words which are mispronounced by the rule-based system, are stored in an exceptions dictionary. It has been estimated that less than 5% of the words in an unrestricted text show some pronunciation defect. This means that the phonetic transcription of less than one word in twenty is not completely correct.

The system incorporates a modest pronunciation dictionary too which can be updated by the user. Such a personal dictionary is very useful to avoid mispronunciations of jargon words, frequently used proper names etc.

It is worth mentioning that the linguistic section of the text-to-speech system was completely developed and implemented using the DEPES development tool.

### *The phonetic section*

The phonetic section of the system performs the segmental and prosodic synthesis. It includes modules to create a correct durational pattern, to synthesize an adequate intonation contour and to create spectral parameters.

The system uses a durational model for Dutch that attributes a well estimated duration to each phoneme. In order to discover durational rules for continuous speech, we analysed the phone durations measured in a Dutch text. This text, with a total duration of more than 8 minutes, was read by one female native speaker of Dutch. The same speaker was used for the development of the segmental synthesis part (see further). Rules were developed to explain the bulk of the durational variations seen in the text. These rules cover durational phenomena such as the short/long opposition of Dutch vowels, word-final lengthening, prepausal lengthening, the influence of prominence etc. The present version of the synthesis system uses a limited set of durational rules which accounts for 79% of the total variance observed in phoneme durations of a text read (VAN COILE, 1987b).

Next, the message to be synthesized is given a correct intonation contour. The system uses a strategy based on the results of a study on Dutch intonation by 'T HART et al. (1973, 1975). The intonation contours are described in terms of a limited set of standardized rises and falls of pitch. A limited number of rules specifies how these elementary pitch movements can be combined to create an intonation contour for an entire message. These rules take into account the number and the location of the dominant words and major syntactic boundaries.

The intonation contour shown in the example of *Fig. 6.* uses only 3 different elementary pitch movements: the rise '1' and the fall 'A' to mark the dominant words, the rise '2' to mark the end of the sentence. Given the description of the intonation contour in terms of elementary pitch movements and the durational information, the fundamental frequency parameter is calculated.

| | |
|---|---|
| GRAPHEMES | "Ik ben een "komputer.<br>(I am a computer.) |
| PHONEMES | #"IG 'bEn Sn kOm-"pju-t$r# |
| INTONATION | 1 A 2 |

*Fig. 6.* Intonation by means of elementary pitch movements.

Finally, segmental synthesis is performed. Our system is based on segment concatenation: an inventory of small speech segments *taken from natural speech* is used to compose an arbitrary synthetic message. An interesting segment type in this respect is the diphone (among others OLIVE, 1977). A diphone is a small speech segment which starts and ends in the stationary parts of two successive phonemes. Thus, the transition between the phonemes is preserved within the segment itself.

In our approach to speech synthesis we use a diphone inventory supplemented by a minority of larger units. The segments are taken from isolated words spoken by a female speaker. The words are analysed by means of a classic LPC-model. Subsequently, the segments are extracted semi-automatically (VAN COILE, 1987c).

Segmental synthesis involves the following steps:

—subdividing the phonetic transcription into segments,

—concatenating the speech parameters of the different segments. These parameters are stored in the segment inventory.

—Setting the estimated duration of each phoneme by stretching and shrinking the time between successive parameter sets.

—Adding the calculated fundamental frequency contour.

The parameters are then used by the speech synthesizer to obtain the acoustic waveform.

## Conclusions

The text-to-speech system just described, has already reached a very satisfactory level of performance. Both the intelligibility and the naturalness of the speech are appreciated by the listener. However, a lot of linguistic knowledge is still missing (e.g. the syntactic analysis is very rudimentary).

A commercial version of the system became available in the fall of 1988.

## References

CARLSON, R.–GRANSTRÖM, B. (1976): A Text-to-Speech System Based Entirely on Rules. *Proc. Int. Conf. Acoust. Speech and Signal Process. ICASSP-76*, Philadelphia, April, 12-14, 1976, pp. 686–688.

HERTZ, S. (1985): From Text to Speech with SRS. *J. Acoust. Soc. Am.*, Vol. 72, No. 4, pp. 1155–1170.

HERTZ, S.–KADIN, J.–KARPLUS, K. (1985): The Delta Rule Development System for Speech Synthesis from Text. *Proc IEEE*, Vol. 73, No. 11, pp. 1589–1601.

OLIVE, J. P. (1977): Rule Synthesis of Speech from Diadic Units. *Proc. Int. Conf. Acoust. Speech Signal Process., ICASSP-77*, New York, pp. 568–570.

'T HART, J.–COHEN, A. (1973): Intonation by Rule: a Perceptual Quest. *Journal of Phonetics* Vol. 1, pp. 309–327.

'T HART, J.–COLLIER, R. (1975): Integrating Different Levels of Intonation Analysis. *Journal of Phonetics*, Vol. 3, pp. 235–255.

VAN COILE, B. (1987a): DEPES: a Development Environment for Pronunciation Expert Systems and its use for Dutch. *Proc. IASTED Conference on Expert Systems*, Genève, June 1987, pp. 122–125.

VAN COILE, B. (1987b): A Model of Phoneme Durations Based on the Analysis of a Read Dutch Text. *Proc. European Conference on Speech Technology*, Edinburgh, September 1987, Vol. 2, pp. 233–236.

VAN COILE, B. (1987c): Computer-Aided Segmentation of Spoken Words Given their Orthographic Representation. *Proc. European Conf. on Speech Technology*, Edinburgh, September 1987, Vol. 1., pp. 277–280.

*Address:*

Bert Van COILE
Laboratorium voor Electronica en Meettechniek
Rijksuniversiteit Gent
St. Pietersnieuwstraat 41, B-9000 Gent, Belgium