

PHONEME TO GRAPHEME CONVERSION BY RULES

L. Boves (*), W. Senders (*), J. Wester (*) & R. Willemse (*)

ABSTRACT

In this paper a method is presented to describe an essentially non-deterministic problem like the translation of phoneme strings into grapheme strings with a deterministic rule-system like the SPE-formalism. The solution lies in the addition of a type-3 grammar and a small post-processor. It is shown that this addition amplifies the power of the SPE-formalism.

INTRODUCTION

In the framework of ESPRIT-project no. 860 'Linguistic Analysis of the European Languages' a linguistic processor is being built that should be usable in several applications in which linguistic knowledge is required. One of these applications is automatic speech recognition. One of the processes that is necessary in systems for large vocabulary speech recognition is some form of sound-to-letter or phoneme-to-grapheme conversion. Even if the phonemic input to the phoneme-to-grapheme conversion process would be completely error free, the output of the process will be ambiguous for most words in most languages. The sub-system that is described in this paper accounts for the translation of one phoneme string into one or more grapheme strings. At present, the sub-system handles only phoneme strings corresponding to words pronounced in isolation. There is, however, no fundamental reason why it could not be extended to connected speech. The overall system chooses, by means of Markov Chains and formal linguistic knowledge, the most likely grapheme string from the cluster of alternative grapheme strings offered by the phoneme to grapheme translation module.

Phoneme-to-grapheme conversion is a string transformation process that seems to have much in common with grapheme-to-phoneme conversion or text-to-speech synthesis, that, in many implementations, has been handled by means of procedures borrowed from phonological theory. Since the publication of Chomsky & Halle's Sound Pattern of English (ref. 1), known as SPE, the formalism introduced in that classic for expressing phonological rules seems to be the preferred choice of most phonologists. Its application is not, however, restricted to phonological rules; rather, it can be applied to a larger class of string transformation problems.

THE SPE-FORMALISM

The formalism for translating strings to strings as it is defined in The Sound Pattern of English is based on string-transforming rules of the Chomsky-1 type (ref. 2). The rules are of the format:

A -> B / C --- D

A: focus
B: structural change
C: left context
D: right context

in which the elements C and D are optional.

(*) Institute of Phonetics, Nijmegen University, The Netherlands

A number of these rules build a grammar. The order in which the rules are assembled in the grammar is crucial, because the rules work upon each others' output. In other words: the first rule of a grammar takes the input of the overall grammar as its input, the second rule, however, takes the output of rule 1. The operation of the last rule yields the transformed string.

PHONEME TO GRAPHEME CONVERSION AND SPE-RULES

Because the notation defined in The Sound Pattern of English has a solid formal basis, it turned out to be possible during the past years to build computer programs that can translate the transformations described with SPE-rules into computer programs (ref. 3) or that can directly execute SPE-rules.

These properties of the SPE-formalism make it an attractive choice for the description of transformations on strings that are necessary when translating phoneme strings into grapheme strings. SPE, and therefore also the computer programs that execute these rules, has one major drawback, however, that poses considerable problems for the translation of phoneme strings into grapheme strings: the rules of a SPE-grammar describe a deterministic process. The translation of phoneme strings into grapheme strings is a non-deterministic problem, however. This seems to be incompatible.

One solution we could imagine is to add a feature to the SPE-formalism that multiplies the number of strings being processed with the number of alternative translations for a specific phoneme. A normal program that executes SPE-rules processes, because of the deterministic character of SPE rules and grammars, only one string at a time. This string is subject to all the changes specified in the rules of a grammar and these rules operate upon this string one after another. If we implement a solution like the above the number of strings being processed can increase significantly, leading to an unacceptable increase of execution time of the program.

PHONEME TO GRAPHEME CONVERSION IN A SEMI-DETERMINISTIC WAY

Our solution to deal with the deterministic character of an SPE-grammar when using them for the translation of phoneme to grapheme strings only uses the given properties of the SPE-formalism without adding anything to the formalism proper.

The SPE-formalism operates upon a predefined character-set. All characters of this set, and of course combinations of these, can be inserted into the string being processed. We can use this property to replace ambiguities that occur on the phoneme level when translating phoneme strings into grapheme strings with a kind of meta-symbols. The meta-symbols then are -for the rest of the rule-set, a non-ambiguous marker for an ambiguity. When doing so the system still can work upon one string.

The string that is output by the program of course still contains these meta-symbols. We can, however, define all meta-symbols in a type-3 grammar and replace them in the output text of the grammar with a simple post-processing program. This post-processing program generates all possible combinations of all alternative translations of all meta-symbols in a string. The control flow in the phoneme-to-grapheme system is shown in fig. 1.

CONCRETE EXAMPLES

The rule-sets that we use in our approach for the translation of a phoneme string into one or more

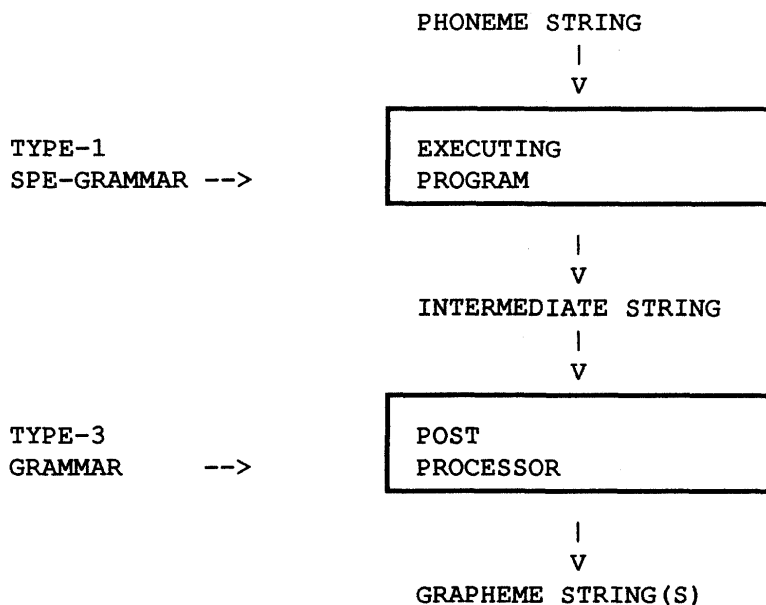


FIGURE 1: Flow of control in the phoneme-to-grapheme system

grapheme strings consist of a type-1 rule-set that generates an intermediate string containing graphemes and meta-symbols, followed by a type-3 grammar that converts the intermediate string to a lattice of word candidates.

As meta-symbol for the indication of an ambiguity on the phoneme level we can, of course, choose any sequence of characters. One of the demands we could make for such a sequence is that it has to be recognisable in an easy way, both for the author of the set of rules and for the contexts of the rules itself. We chose sequences of the format:

(12)

in which the number '12' represents the type of ambiguity and the brackets are for the isolation of this number of its environment. When doing so we can make rules that generalise over meta-symbols as is shown in the following example:

$*(15^*)*(13^*) \text{ --> } de / \# \text{ ---}$

This rule replaces the chain of two meta-symbols '(15)(13)' by the grapheme string 'de' if it occurs at the beginning of a word: '# ---'.

Two examples of SPE-rules that replace an ambiguity on the phoneme level by a meta-symbol are:

$@ \text{ --> } *(13^*) / \neg\{nj/pj/tj/kj/sch\}\neg \text{ --- } \#$

$f \text{ --> } *(9^*) / \{f/p/t/s/g/k\} \text{ --- } \{[+voc] / r/l\}$

The first rule means: replace an '@' by the meta-symbol (13) (the asterisks are a convention of the program we use for the compilation of the SPE-rules) when this '@' is not preceded by one of the elements of the list 'nj', 'pj', 'tj', 'kj' or 'sch' and is followed by the end of the word. This rule yields the following translations:

del@ (to share) ==> del(13)
 b@del@ (to dole out) ==> b@del(13)
 bompj@ (small tree) ==> bompj@

The second rule means: replace an 'f' by the meta-symbol (9) if it is preceded by one element of the list 'f', 'p', 't', 's', 'g' or 'k' and is followed by either a phoneme that can be characterised by [+voc] or by an 'r' or an 'l'. This rule yields the following translations:

Atfizer@ (to advise) ==> At(9)izer@
 sXepfart (navigation) ==> sXep(9)art

The '@' at the end of the phoneme string 'At(9)izer@' matches the context for the first rule. After the two rules have operated, the grammar yields the following string:

Atfizer@ (to advise) ==> At(9)izer(13)

The two ambiguities we introduced above, (9) and (13), are written out in the type-3 grammar as follows:

9: f; v.
 13: e; en.

The output of the total system when feeded with the input-string 'Atfizer@' is:

atfisere
 atfiseren
 atvisere
 atviseren

CONCLUSION

The translation of phoneme strings into grapheme strings in the way we propose in this paper has a number of advantages. These advantages are:

1. We can also use the well defined and powerful SPE-formalism for the translation of phoneme strings into grapheme strings by only adding a type-3 grammar;
2. We can use one of the existing programs that translate SPE-rules into a computer program;
3. We postpone the copying of the processed string into many processed strings as long as possible. At the moment we duplicate the processed string no string transformations have to be performed anymore. This means a significant gain in processing speed.
4. By using meta-symbols we introduce extra power in the SPE-formalism: we can now make rules that act upon meta-symbols or combinations of meta-symbols. This was demonstrated in an example above.

REFERENCES

1. N. Chomsky & M. Halle, **The Sound Pattern of English**, 1968
2. N. Chomsky, 'On certain formal properties of grammars', in: **Information & Control** 2, pp. 137-167, 1959.
3. J. Kerkhoff, J. Wester & L. Boves, 'A compiler for implementing the linguistic phase of a text-to-speech conversion system', in: H. Bennis & W.U.S. van Lessen Kloeke (eds.), **Linguistics in the Netherlands 1984**.