A GRAPH-ORIENTED APPROACH TO THE GRAPHEME-TO-PHONEME TRANSCRIPTION OF ITALIAN WRITTEN TEXTS.

P. COSI*.  

ABSTRACT

Taking into account the implicit phonotactic constraints of the Italian, a graph-oriented computational approach to the grapheme-to-phoneme transcription of written texts is described. The system is based on the mathematical theory of Finite States Automata, generalized and augmented to obtain a simple syntax-directed translation schema. The rules are totally based on the pure orthographic form of the words, and no level of grammatical knowledge of their classes is considered. A particular set of exceptions is often associated to the whole set of rules and, owing to the modular quality of the automaton, an optimized rule-exceptions check mechanism is built up, albeit in a preliminary and incomplete form.

INTRODUCTION

Computers cannot synthesize reasonable-quality sentences from phonemic information alone because the acoustic features of human utterances are heavily influenced by syntactic, semantic and pragmatic factors (ref 1). However, if we focus our attention on speech synthesis from written texts, the first problem encountered is that of grapheme-to-phoneme conversion. This paper describes a graph-oriented transcriber, based on the theory of Finite States Automata (ref 2), which should be considered as the basic component of a more complex synthesis-by-rule program.

OUTLINE OF THE GRAPHEME-TO-PHONEME TRANSCRIPTION RULE SYSTEM

The whole set of transcription rules for Italian (ref 3) is entirely based on pure orthographic forms. The automaton is supposed to accept stressed words as input, so a preliminary analysis of the automatic assignment of lexical stress has to be considered (ref 4). The graphemic input words are indicated by capital letters to distinguish them from the outputted phonemic words which follow the International Phonetic Alphabet (IPA). The transcription procedure is divided in two levels. The first one (T1) corresponds to a "broad" or "phonemic" transcription, while the second is closer to an "allophonic" or "phonetic" conversion (T2). In T2, three allophones of n (/n/, /RN/, /n/), two semivowels /i/ and /u/ (to distinguish from the semiconsonants /j/ and /w/) and long stressed vowels (the vowel's symbol is followed by a semicolon) are added. The stress mark (') precedes a stressed syllable in T1 only, while in T2 it is directly connected

*Centro di Studio per le Ricerche di Fonetica (C.N.R.).  
Via G. Oberdan, 10 - 35122 Padova (Italy)
to the corresponding stressed vowel (•); moreover the geminated consonants are indicated by doubling their symbols in T₁, while in T₂ they are represented by the same lengthening mark used for long stressed vowels. The most commonly utilized formalism describing the rules is the so-called context-dependent one (ref 5). A rule is indicated in the following way:

\[ W \rightarrow X / Y \_ Z \] (1)

which means that W becomes X if it appears between Y and Z. W, Y and Z are elements belonging to the input alphabet (i.e. graphemes), and X is an element belonging to the output alphabet (i.e. phoneme). In Table 1, the whole set of "first-level" rules and the "second-level" rule referring to grapheme N are formalized context-dependently. To produce a non-ambiguous grammar it is necessary to introduce new rules for the graphemes S and Z to cover all the permissible Italian phonotactic possibilities. For the time being, two maverick phonemes (s) and (z) are added in the cases not taken into consideration by the rules expounded so far.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Context</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>A /a/</td>
<td>N</td>
</tr>
<tr>
<td>2)</td>
<td>E /ō/</td>
<td>[H]NTO</td>
</tr>
<tr>
<td>3)</td>
<td>I /u/</td>
<td>[θ]N</td>
</tr>
<tr>
<td>4)</td>
<td>S /I/</td>
<td>/aC/II</td>
</tr>
<tr>
<td>5)</td>
<td>GL /l/</td>
<td>N</td>
</tr>
<tr>
<td>6)</td>
<td>GN /n/</td>
<td>/θ/</td>
</tr>
<tr>
<td>7)</td>
<td>H /f/</td>
<td>N</td>
</tr>
<tr>
<td>8)</td>
<td>Z /s/</td>
<td>/θ/</td>
</tr>
<tr>
<td>9)</td>
<td>F /j/</td>
<td>N</td>
</tr>
</tbody>
</table>

SYMBOLS:
- logical or ( ) optional
- parallel choice

Table 1. Context-dependent formalization of the transcription rule system.

**GRAPH-ORIENTED IMPLEMENTATION**

The transcriber is seen as a set of several automata grouped together to form a complex deterministic finite state automaton which is conveniently represented by its "state transition diagram" (a directed graph with nodes for the states connected by arcs labelled with input symbols causing transition and possibly with output symbols caused by the transitions). The graph is "augmented" by controls and actions which can arise depending on the different situations within the "walk" in the graph. The mechanism of the transcription can be considered as that of walking across the graph from an initial to a final state, and the route followed directly depends
on the different types of nodes and situations in the graph.

All the rules are organized in order to take into account only the context on the right. In fact, the graphemic input string (IS) is inspected while it reads its symbols on the right of the current pointer position which is continuously increased, depending on the successively "eaten" symbols.

Figure 1 shows the graph's block diagram. Apart from the word-final-symbol rule (the walk reaching the final node), all the rules are represented by a walk ending in the activating node (AC). The word-initial-symbol rule (some rules consider word initial conditions and should be treated in a different way from the other rules in a left-to-right approach) can easily be recognised as the walk connecting the initial node (IN) with AC, while the other word-embedded-symbol rules (WESR) are represented by the graphs which start and end in AC. Thus AC is the only node which has an output string (OS) connected to its input links, a feature which means that a rule has been tested and applied. All the nodes are represented as shown in Fig. 2 and are defined as "reading" (m>0) and "non-reading" (m=0). They differ because the former need to read m IS symbols (X_i) to proceed in the graph, while the latter only need to utilize and possibly "consume" symbols previously read. When a rule is activated, the symbols which the corresponding module needs to verify are read by the first reading node (see rule-nodes or R_x in Fig. 1, referring to WESR). These symbols are successively utilized by the next non-reading nodes of the automaton until the whole set of alternatives is examined to complete the rule. This means outputting a string (Y_i), repositioning the IS pointer (n_i represents the "backtraking" IS output pointer, i.e. the difference between the number of IS symbols utilized by the rule and the number of those effectively transcribed) and, finally, restarting the procedure with a return to AC.

The exceptions control block (ECB) in Fig. 2 is very important and is used to verify whether the current input word forms part of the exceptions to the current applied rule, in order to transcribe it by table look-up.

Two examples of sub-automata are given in Fig. 4. The first implements the word-initial-symbol rule (part of rules 4, 5, 8 in Table 1) together with some exceptions...
syllable of a word begins (rule 8) with a voiceless consonant. The second refers to grapheme B. Finally, Table 2 illustrates some results of the transcription procedure applied to a reduced set of Italian words.

Fig. 4. Two automaton examples. Table 2. Some Italian words.

CONCLUSION

An advantage of the graph-oriented approach to phonemic transcription is its relative simplicity of implementation owing to the "modular" nature of the complete automaton. Each subgraph corresponds to one particular subset of the overall rule system and can be easily tested, independently of the rest of the total automaton. Another advantage is that it implements an "optimized" rule exceptions verification mechanism albeit in a preliminary form. In fact, it is possible to connect an exceptions file, to each "rule node" in the graph, to each "subrule node" or, indeed, to any node, so breaking down the mass storage memory required to contain the exceptions and reducing the number of those which will be checked walking across the graph, evidently speeding up the transcription procedure.

REFERENCES