COMPOST: A RULE-COMPILER FOR SPEECH SYNTHESIS

G. BAILLY, A. TRAN

ABSTRACT

COMPOST is the result of our first reflections on the computing requirements for synthetic speech generation. It combines the basic facilities of a rule-oriented software with object-oriented design. COMPOST is a tree compiler: it manipulates and transforms not only lists of atoms but also trees and subtrees. Basic atoms involved in the tree inherit of properties of user defined classes. Features and numerical cues are associated with each atom. COMPOST has then numerical capabilities to do synthesis-by-rule.

COMPOST has been developed to build a formant-based rule system but may generate any kind of parametric trajectories and then could be used in rule-based articulatory synthesis.

I. INTRODUCTION

High quality speech synthesis involves cooperation of multiple experts: linguists, phonologists, phoneticians. Computer programming must not mask the underlying knowledge these experts have put into the software. Flexible software environments have already been proposed to offer powerful knowledge representation languages [4,6,7,9,12].

If early languages focused on numerical capabilities (definition of contours, acoustic targets manipulations...) for parametric generation of speech, recent systems like DELTA, DEFES and SCYLA focus on symbolic manipulations with a common multilevel representation of the working data structure.

These interests reflect the three major requirements for a "speech compiler": a) structural multi-layered representation of the data structures, b) symbolic manipulation of data, c) numerical capabilities for effective generation of parametric trajectories.

We have described elsewhere our approach to speech synthesis [2] which is likely to the one generally adopted in Computer Assisted Translation of text [5]: the translation operates in 3 steps: a) the analysis, b) the transfer and c) the generation. In our case the analysis consists of building a structural representation of the linguistic content of a text element (often sentence by sentence): the result is often a tree representation or its projection as a DELTA structure. The transfer consists of replacing the formal description of the text by the formal description of the message: this includes grapheme-to-phoneme transcription and prosodic structure generation. Of course this could not be achieved without information issued by the previous analysis step. Generation then consists to project the formal representation of the message onto the time domain using adequate parametric representations (formants, LPC coefficients, articulatory parameters...).

COMPOST aims to give software support for the two last steps b) and c) leaving text analysis to more powerful software (morphological decomposition using dictionaries and finite state automata, hidden markov models, context free grammars...).

II. THE COMPOST LANGUAGE

The classical well known formalism used in generative phonology is used to express COMPOST rules:

Rule: A -> B/C+D;

which means that the focus pattern A is changed into the transformed pattern B if the left context C and the right context D are verified.

An ordered set of these rules is called a grammar. A COMPOST program consists of a declaration part and a list of grammars.

II.a. The basic data structure

The rules operate on a current representation of the text currently analyzed. This representation is a tree structure which is also the input and output data structure. Fig.1 presents an example of an input sentence in the alphanumeric COMPOST notation: brackets indicate the tree structure, angle brackets enable the user to precise features and/or cues which are not in the standard definition of an object.

II.a.1. Input and output representation.

COMPOST have to be fed with normalized text
input and outputs normalized text. The format of the text follows the same format as the transformed pattern: tree levels are specified by parenthesis and non standard feature and cue values could be added to an object by angle bracketing as in Fig.1.

\[
\text{Det}(\langle L, [maj] \rangle E) \text{ Adq(PESSI) } \langle \text{Noun, [mas, sng]} \rangle (\text{CHAT}).
\]

Fig.1: Tree representation involving Word and Graph levels.

Yet the flow of our French text-to-speech system enters twice into COMPOST. Our strategy consists into the following: a) first, the orthographic sentence is fed into a lexical analyzer using a large dictionary and a morphological grammar of French, b) the output is fed into a COMPOST grammar in order to use word endings, grammatical variables and lexical context to solve some lexical ambiguities and unknown classes, c) the output still full of ambiguities is fed into a HMM lingustic filter and prosodic parser (we are currently studying a neural network based parser), d) finally, the output of this filter is fed back into COMPOST to achieve grapheme-to-phoneme transcription, prosodic structure building and parametric generation.

Although we have proposed in 1986 a pre-syntactic analysis using only contextual information \([1]\) to solve lexical ambiguities and insert prosodic markers, we do not think it is sufficient to label correctly unlimited text as mentioned in \([10,11]\) where more robust syntactic information is needed.

II.a.2. Declaration of objects.

Basic atoms of this structure are defined in the declaration part. These atoms are instances of generic objects which inherits properties of atom classes. The declaration part consists of declaring the classes, the different atoms they group as well as associated features and cues. Cues are numerical values which are attached to each instance of an object.

At each class definition, default features and cues for objects may be precised. Fig.2 shows the definition of different classes of objects. Parametric generation presented in the following imposes the existence of two classes and two cues: Phoneme with the cue duration and Target with the cue Instant (cf. II.I.c). These predefined names are necessary to project and synchronize the Target objects onto the time axis.

II.b. Writing rules and grammars.

Grammers are applied sequentially on an input tree structure and each application of one grammar produce an output tree structure which becomes the input tree for the next. Grammars may be applied recursively (the header of the grammar gives the minimum and the maximum number of times it will be executed (see Fig.3). The rules are executed following the next rules for pattern matching: a) the rule with the longest matching focus pattern is examined first, b) in case of equal length, the rules are examined following the order of declaration.

Patterns and contexts of the rules must be subtrees (only one class can be present at one level). Focus patterns and contexts may contain imprecise number of objects: minimum and maximum number of occurrences of these objects may be precised with square brackets as in:

\[
\text{Num1: 1} \\
\rightarrow \langle \text{Co}(\epsilon) \text{ Nb(o-2)} \text{ Nb(mil)} \text{ Nb(7,9)+[num]*[3]} \rangle \\
\rightarrow \langle \text{Co}(\epsilon) \text{ Nb(x-)} \text{ Nb(mil)} \text{ Nb(7,9)+[num]*[3]} \rangle
\]

which describes part of the pronunciation of the grapheme "1" for French.

II.b.1. Modifying objects, features and cues.

Of course the rule writer may assign, or subtract features of existing objects, assign cues or change the object’s name. This is used in the following example to guess unknown lexical classes which have been evidenced by the lexical analyzer by the word endings i.e. every word ending with ATION is a noun:

\[
\text{Rg: Inc} \rightarrow \langle #1=\text{Noun, [mas, sng]} \rangle / + ([let]*[1,20] \text{ATION})
\]

II.b.2. Symbolic variables.

Rules may transfer and/or duplicate atoms or whole subtrees from the focus pattern to the destination pattern by two ways: symbolic variables or an index number variable.
Symbolic variables store any subtree which match against the focus pattern including it. For example, the following rule treats the prefixed English notation for units by means of a symbolic variable #A:

\[ \text{Del: } \text{Noun}($)$\text{Nb}([num]*[1,10]) \rightarrow \text{Nb}([#A]) \text{Noun}(dolar);\]

The same rule could be written using an index number variable #4, which means "take the 4th atom of the focus pattern":

\[ \text{Del: } \text{Noun}($)$\text{Nb}([num]*[1,10]) \rightarrow \text{Nb}([#4]) \text{Noun}(dolar);\]

II.b.3. Numeric variables.

Rules may assign expressions to cues. These expressions may contain any other cue of the concerned object or local variables which have memorized values of cues of objects of the focus pattern in a way similar the KTH system. The following example calculates the duration of a word:

\[ \text{grammar Init}[1,1] \]
\[ \text{Init: } \text{phon} \rightarrow <#1,ok=0>; \]
\[ \text{endgrammar} \]
\[ \text{grammar Count}[1,50] \]
\[ \text{Count: } \text{Word} <\text{phon},ok=0>|[0,10] \text{<phon},ok=0,d=duration> \rightarrow <#1,\text{dur}+=\text{dur}> \quad (#2 <#3,ok=1>); \]
\[ \text{endgrammar} \]

An example is given in Figs. 4a. and 4b.: this shows a simple example of an /i-y/ transition on the three formants involving formant crossing.

\[ <\text{duration}=100> \quad (\text{<step,F1}=300,\text{F2}=2000,\text{F3}=2600,\text{Instant}=0>) \quad (\text{<line,F3}=2600,\text{Instant}=50>) \quad (\text{<step,F3}=1800,\text{Instant}=20>) \]

An example is given in Figs. 4a. and 4b.: this shows a simple example of an /i-y/ transition on the three formants involving formant crossing.

\[ <\text{duration}=150> \quad (\text{<step,F1}=300,\text{F2}=2000,\text{F3}=2600,\text{Instant}=0>) \]

II.c. Synthesis-by-rule and generation of parametric trajectories with COMPOST.

If symbolic manipulation of objects or structure is sufficient to write linguistic rule components, the reader of certain articles in the references has the feeling that something is missing to achieve the effective generation of frames of parameters able to feed the chosen synthesizer. Of course if the rule-writer may call at every moment a user-defined routine, this objection doesn't make sense. We do think that the most important feature of a rule-based system is its coherence. A unified formalism enables debugging facilities while preserving the easy reading of the synthesis strategy.

Our strategy for parametric generation is compatible with the COMPOST language and structure. The first complete text-to-speech system we realized was based on a diphone technique [3] with formant synthesis. This diphone dictionary is often sufficient to realize good formantic transitions and part of it could be used to ground a rule-based system. An easy way to switch between a dictionary-based generation to a rule-based system consists of a necessary step of stylization [8]. Stylization of natural data according to a precise model provides the operator with a small number of commands he may manipulate with rule-based systems.

Main advantage of this approach is that stylization of natural data could be compared to synthetic one, heard and seen with the REGLES software in a full compatible way.

II.c.1. Specifying parametric targets.

Our model of parametric stylization is very similar to the KTH system. We have generalized the notion of targets and transitions functions: COMPOST imposes the existence of the class Target. The type of transition from a Target object to the next Target object is specified with the first object's name. The first cues of the class Target (until the definition of the cue Instant, see below) are the parameters the COMPOST program has to generate. Values of the targets are then given by the values of the cues. Only the cues whose values are different of a minimum value (specified in the declaration part) are concerned by a given target.

II.c.2. Specifying timing of parameters targets.

In order to position targets onto the absolute time axis we have the following strategy : the class Target must have the cue Instant. This cue expresses the relative timing of the target to the duration of its dominating phoneme. This cue is referenced to the beginning of the dominating Phoneme. This cue Instant could be negative or superior to 100 : the transition could begin before or after the arbitrary phoneme limits. Each Phoneme may dominate any number of Targets.

An example is given in Figs. 4a. and 4b.: the duration of a word:

\[ <\text{duration}=100> \quad (\text{<step,F1}=300,\text{F2}=2000,\text{F3}=2600,\text{Instant}=0>) \quad (\text{<line,F3}=2600,\text{Instant}=50>) \]

\[ <\text{duration}=150> \quad (\text{<step,F1}=300,\text{F2}=2000,\text{F3}=2600,\text{Instant}=0>) \quad (\text{<step,F3}=1800,\text{Instant}=20>) \]

II.c.3. Modeling coarticulation by rules.

The rules may change spatial as well as time position of the Targets by assigning new values to cues corresponding to
parameters or the cue Instant. By means of the unlimited number of Target dependents of a given phoneme, complex contours could be defined. These contours, their shapes could be changed using context-dependent rules of the COMPOST language.

As mentioned in the following paragraph, COMPOST has been coupled with a graphic editor called REGLES which enables the operator to change by means of a mouse the targets and their position by hand. The trajectories could be visualized in different views: superposed trajectories, added trajectories... This visualization software is coupled with the synthesizer: each satisfactory synthesis obtained by hand could be produced by rules.

III. THE COMPOST ENVIRONMENT

There are only two components of COMPOST: the pre-compiler and the visualization software REGLES using multi-windowing techniques: the pre-compiler generates from a declaration file and grammar files a C file containing tables initialization. The COMPOST analyzer is obtained by compiling this file with a conventional C compiler and linking the result with a COMPOST library. Only some declaration of REGLES need to be modified (association of the COMPOST Target objects with windows and mode of visualization).

COMPOST and REGLES are now running on a PC. All software in written in C and can be easily transferred to another computer. COMPOST has simple debugging facilities: the trace option lists rules applied with explicit patterns and contexts as well as the resulting current structure.

We are currently adding to the REGLES software simple signal processing such as temporal and sonagraphic representation of the synthetic speech.

IV. CONCLUSIONS

The COMPOST system provides flexibility and coherence to rule-writers of text-to-speech systems. Symbolic and numerical capabilities of this knowledge representation will easy the development of a complete text-to-speech system for French by rules using formantic trajectories.

We are planning to add predefined functions like counting the number of dependents of a certain type of a certain object or more complex procedures. But we will limit these extensions to a minimum in order to maintain coherence of the strategy.

We have already tried a multimedia synthesis by generating in the same COMPOST program trajectories for acoustic parameters and articulatory parameters to drive a simplified model of the lip contours. REGLES then produce a simultaneous a synthetic acoustic output and an animated lip contour.

BIBLIOGRAPHY