Rule extraction for allophone synthesis

Institute for Perception Research
PO Box 513, 5600 MB Eindhoven

LtB/ltb 91/04
15.02.1991

Rapport no. 783

Rule extraction for allophone synthesis

Final report ALLODIF

L.F.M. ten Bosch

RULE EXTRACTION FOR ALLOPHONE SYNTHESIS

Final report ALLODIF

Louis ten Bosch

Contents

1 Summary 3

2 Introduction 4

3 The diphone data set and the allophone rule set 6

3.1 The data set 6

3.2 The rule set 6

4 Diphone preprocessing 11

4.1 Diphone stylization. 11

4.2 A preliminary perception experiment 14

4.3 The choice of the diphone set 14

5 Automatic rule extraction 16

5.1 The conversion from phonological rules to numerical rules and vice versa. 17

5.2 Algorithmic search for numerical rules on a database. 19

5.3 Example: the l-rule 23

6 Perceptual relevance of rules 27

7 Application in rule sets 29

8 Discussion 33

A The Eindhoven en Nijmegen parameter set 37

B The Nijmegen feature specification 38

C The covariance method 40

D A rule interpreter 43

1 Summary

The Dutch SPIN-project Analysis and Synthesis of Speech ASSP was initiated in December 1985. In this project, linguistic, acoustic and signal-analytic studies aimed at a better understanding of the text-to-speech process for Dutch.
This report decribes the results in the SPIN-ASSP project ALLODIF. This project aims at the (automatic) extraction of allophone rules from diphone speech data for Dutch. The project was initiated in March 1988, and ended on 31 December 1990. It was set up with the idea that the development of the *allophone synthesis* could profit by the (at that time) higher intelligibility and naturalness of the *diphone synthesis*. Rules for the Dutch allophone system are developed in the SPIN-ASSP project ALSYS (by Henk Loman) at Nijmegen University, whereas the diphone system is being developed at the Institute for Perception Research (IPO), Eindhoven.

One of the main findings of this project is, that the automatic extraction of allophone rules is possible under specific conditions but that it is not as straightforward as was expected initially. Although this project does not produce a set of rules in the format of the Nijmegen rule set, several techniques were developed to tackle the problem from different points of view. These methods may become of interest as soon as large speech databases and fast computing facilities are available.

In this report, we will first discuss the background of the problem (section 2), and the data we have used: the Eindhoven diphone data and the Nijmegen rule set (section 3). After that, the approach in ALLODIF is dealt with in detail. In section 4, we will deal with the required preprocessing of the diphone set.
Sections 1 to 4 are preliminary. The actual method for the extraction of rules will be discussed in section 5 and further. A brief remark on perceptual evaluation is made in section 6. In section 7, the question of the optimization of rule *sequences* is discussed. The final discussion can be found in section 8.

In this final report, we will not go into technical details. These details will be described in two other reports, one dealing with the approach in ALLODIF but at a technical level, the other with the implementation of programs and algorithms. These reports are in preparation and will be soon available from the author. This final report is only meant to illustrate the approach in the project ALLODIF.

2 Introduction

From speech perception experiments, it is well-known that the proper modelling of speech *transients* is crucial for the synthesis of intelligible and natural speech. Transients, which correspond to the dynamic events in the speech signal, result mainly from the coarticulation process. In principle, transients are determined by a number of factors: phonetic context, speaker, speaking rate, and speaking style. If we aim at speech synthesis representing one speaker and one speaking style, the transients are determined by context and speaking rate and they can be modelled by a proper collection of context-dependent rules. In principle, these rules can be found by analyses of natural speech. Unfortunately, this is not a trivial task. Dennis Klatt spent more than 15 years to produce his version of Klatt-talk (now available as DEC-talk).

Diphone synthesis

The problem of designing rules can be avoided by using segment-based speech synthesis, such as diphone synthesis. In this method, the speech signal results after concatenation of *speech segments* that are segmented from natural utterances. Segments are constructed to contain the transients between phonemic targets, or other problematic sound sequences such as consonant clusters. For this reason, the segment synthesis is rather succesful without much effort, and that is its major advantage. What we need is a proper segmentation algorithm and a concatenation algorithm. (For simplicity, we do not deal now with the required prosodic structure.) A disadvantage is, that we cannot manipulate the resulting speech to the extent we want, as all segments are principally fixed. When we need another voice, e.g. to simulate dialogues, we in general have to construct a completely new segment inventory.
At IPO, speech synthesis is based on diphones (cf. Elsendoorn & 't Hart, 1982). A diphone contains a sound sequence from the 'midpoint' of a phoneme to the 'midpoint' of the next phoneme. Longer segments, such as syllables or demi-syllables, and shorter segments, such as sub-phonemic speech units, may be used as well (cf. Olive, 1990).

Allophone synthesis

Another speech synthesis method is based upon *rules*. Here, speech is considered to be a sequence of allophones, of which the speech parameters are to be specified explicitly. These speech parameters result from a *table*, containing the default values for all speech parameters for each allophone, and from *rules*, in which the parameters are modified according to context. A major advantage of the rule-based speech synthesis is the parametric freedom: within the limits of the applied synthesizer there is no restriction on the complexity of the output. In principle, rule-based synthesis can sound very intelligible and natural: DEC-talk provides a good example. A disadvantage

is that the definition of rules may be a tedious task. Rules often result by a trial-and-error process.
Examples of allophone synthesis are provided by e.g. the Dutch SPIN-project ALSYS (carried out by H. Loman, in Nijmegen; cf. Loman, Kerkhoff & Boves, 1989), by MI-talk (for American English, Allen *et al.*, 1987), and in the Hungarian MULTIVOX-system (Olaszy, Gordos & Nemeth, 1990).

ALLODIF was set up as a bridging project between these two methods for speech synthesis. In 1988, when the project ALLODIF started, the segmental speech quality of the Nijmegen allophone system was surpassed by the quality of the Eindhoven diphone system (van Bezooijen, 1990). At that time, the question was posed how rule improvement might be helped by careful analyses of the diphone set. The concrete problem became, how to extract allophone rules from diphone data, or, in general, from natural speech data. Problems of this kind appear in many fields, where systematic regularities are to be derived from potentially blurred measurements.

In the project we have shown that the problem is solvable, but it takes time and there are some restrictions. We give a list of them:

1. we have to define the *kind of rules* we want to extract (the model);
2. we must indicate in what detail the *data* are to be represented;
3. we must consider the problem of the *perceptual relevance* of rules;
4. finally, we face the problem of the *different characteristics* of the Nijmegen and Eindhoven synthesisers.

In this report, we will deal with these points in detail. In order to be able to relate the data on the one hand and rules on the other, we first consider the available diphone data set and the Nijmegen rule set (next section). After that, we discuss the rule-extraction problem.

3 The diphone data set and the allophone rule set

3.1 The data set

The Eindhoven diphone set used for rule extraction purposes contains about 1500 diphones. These diphones were segmented from stressed syllables in utterances spoken by the Dutch speaker H. Zelle. The Zelle-set is often used in the diphone synthesis system at IPO. One diphone is characterized by the behaviour over time of a specific set of speech parameters. The parameter values are stored in *frames* which are updated every 10 ms. The twelve most important parameters are the five formant frequencies, the five bandwidths, the energy, and a voicing parameter. In the sequel, we will concentrate on these parameters. Other parameters, of secondary importance, regulate e.g. the pre-emphasis and the frame duration.

Rules constructed to simulate diphone data must prescribe all these parameter tracks except for perceptually irrelevant details. At first sight, that implies that one diphone can be characterized by a set of twelve rules for every of the twelve tracks. We will see, however, that this approach does not necessarily lead to optimal rules.

3.2 The rule set

Since we aim to construct a tool for the allophone rule development, we now consider the Nijmegen rule set in some detail. The essential feature of an allophone rule is its transformation ('mapping') of a linear input string into a list of target values for synthesis parameters.
Normally, the rules are sequentially ordered in one file. The present Nijmegen file (version dated October 1990) contains 2059 lines.

A *rule* consists of a focus specification, a context specification and an action. Its input is a string of symbols. These symbols represent either (sub)phonemic acoustic targets or supra-segmental (higher-level) information. An example of an input string[1] is

```
oppPal
```

representing the Dutch word 'opaal'. The string 'ppP' is mapped by the rules to a sequence of three acoustic target frames representing silence, burst and aspirated part, respectively.

[1]This example does not strictly follow the Nijmegen conventions. It is meant as an illustration only.

A rule set may contain several hundreds of rules. An exact figure cannot be given as almost all rules are compound rules, viz. perform more than one action in the same context. In the Nijmegen set, there exist 464 context specifications with a total of 1139 speech parameter assignments; these numbers give an impression of the complexity of this rule set. More accurate complexity measures, e.g. the mean number of contexts fitting per segment and assignments per segment in an 'average' Dutch sentence, are not known.

Almost all rules make use of an SPE-like format (Chomsky & Halle, 1968). Here we show a small part concerning the /m/:

```
m ->>       1 FORM1  := 130 / {y/U/u/@} [nonseg]0 ---
m ->>       2 EQFORM2  := -1
              FORM2 -:= 200 / {&/A/O/a/o} [nonseg]0 ---
m ->>       3 EQFORM2  := -1
              FORM2 -:= 50
              FORM3  := 2100 / u [nonseg]0 ---
m ->>       1 FORM3  := 2100 / @ [nonseg]0 ---
```

/m/ before the assignment symbol

```
->>
```

stands in the focus position. After the assignment symbol, an integer denotes the number of numerical assignments to come. Between the focus and '/' the action is indicated. After the '/', the context is specified. The

```
---
```

denotes the position of the focus in the context specification. For example, the first rule fits as soon as /m/ is preceeded by any of the vowels between '{' and '}' in

```
{y/U/u/@}
```

Since the Nijmegen rule set has as its input a linear symbol string, the context specification is enriched by non-segmental, higher level information. In the example, the set

```
[nonseg]
```

denotes the set of all non-segmental symbols.
The context specification

```
u [nonseg]0 ---
```

means that non-segmental characters *may* occur between /u/ and the focus.

In the first rule, the action is of the form $F_1 := 130$. This means that, if the rule fits, the target value of the first frequency is set to 130 Hz. This action is of the general form $p(\phi) := c$ where ϕ denotes the segment (here /m/), p is a parameter (here F_1) and c is a constant (here 130). This is only one type of assignment. If one has a close look at the Nijmegen set, four different types of assignment can be distinguished. In the third line, we see:

```
FORM2 -:= 200 / ...
```

which says $F_2 := F_2 - 200$. This rule is of the form $p(\phi) := p(\phi) + c$, with $c = -200$.
A third type is present in the second line of the example:

```
EQFORM2 := -1 / ...
```

This action equals the F_2 of the current focus (here /m/) to the *table value* of F_2 of the preceding segment. The value -1 points to the preceding segment. Other values are possible as well. In the action:

```
EQFORM2 := 1 / ...
```

F_2 is taken from the *table value* of the *next* phoneme. A value 2 or -2 points to the next-neighbouring phoneme, et cetera.
All actions of type 3 are of the form $p(\phi_1) := p(\phi_2)$, in which ϕ_1 and ϕ_2 are neighbouring or next-neighbouring segments. Here, $\phi_1 = $ /m/; ϕ_2 apparently depends on the context.
The fourth and last type occurs in the following example, regulating the vowels before /h/:

```
   [+voc] ->>  4 SET Z1=F1
                 SET Z3=F4
                 BANDBRZ1 := 2000
                 BANDBRZ3 := 2000/ --- [nonseg]0 h
```

The SET command equals the frequencies of the first and third *zeroes* Z_1 and Z_3 to the *current values* of the first and fourth *formant* frequencies F_1 and F_4. These actions are of the type $p(\phi) := p'(\phi)$, where e.g. $p = Z_1$ and $p' = F_1$, and ϕ vocalic before '[nonseg]0 h'.

We summarize these types of action as follows:

type	assignment			description	right-hand side
1	$p(\phi)$	$:=$	c	simple	
2	$p(\phi)$	$:=$	$p(\phi)+c$	simple	current
3	$p(\phi_1)$	$:=$	$p(\phi_2)$	'horizontal'	table
4	$p(\phi)$	$:=$	$p'(\phi)$	'vertical'	current

We can order the actions with respect to the difficulty with which they can be found by an algorithm. The simple assignments of type 1 and 2 turn out to be most easy to find. Type 1 actions are simple redefinitions. Of all the 1139 assignments, 611 (about 54 %) are of type 1, while 349 assignments (31 %) are of type 2. We will see in section 5 that between actions of type 1 and of type 2, there is no substantial difference.

The actions of the third and fourth type, which I call the horizontal and vertical assignment for apparent reason, are more difficult to extract than the rules of type 1 and 2. The actions of the third type belong to the next difficult class, whereas type 4 represents the most difficult class. Fortunately, they occur less often than those of type 1 or 2. Type 3 occurs 111 times, type 4 occurs 68 times. Actions of type 3 are introduced for the purpose of special effects, such as the sudden introduction and disappearance of nasality, and for ruling the fricatives. Type 4 actions occur at plosives, fricatives, and in the rules for vowels in their neighboorhood.
In actions of type 3, the assignment in the Nijmegen rules at the right-hand side is done by table values, instead of by actual parameter values. These actions *can* be extracted algorithmically from speech material as well, due to the fact that, in turn, table values can always be written as output of polynomial functions with phonological features as arguments[2]. From the algorithmic point of view, type 3 actions represent a roundabout way. Therefore, this action type is not strictly necessary in a synthesizer rule set.
In rules of type 4, the parameter modification may depend on phonetic parameters that *may have been* altered by former rules. As the history of parameter values is unknown, these actions are the most difficult to extract.

The statistics of all the types are shown in the following table.

type	description	occurrence
1	simple	611
2	simple	349
3	horizontal	111
4	vertical	68
total		1139

[2]This property holds since in principle the set of phonological features discriminates the set of segments: no two segments exist with exactly the same feature specifications. In the Nijmegen feature value specifications, this is not established precisely.

We observe that in the first example dealing with the /m/, two action types are combined:

```
m ->>          2 EQFORM2 := -1
                 FORM2 -:= 200 / {&/A/O/a/o} [nonseg]0 ---
```

Here, first the F_2 of the /m/ is set equal to the F_2 of the preceding vowel (action of type 3); then, F_2 is decreased by 200 Hz (type 2). Even 'diagonal assignments' with the elaborate form $p(\phi_1) := p'(\phi_2) + c$ are decomposable into assignments of the more simple types 1 to 4.

As the above four action types structurally cover the Nijmegen rule set, it is of importance to look for methods to extract them from speech data. In order to do so, we first preprocess the diphones such that the rules can be derived more easily. This preprocessing method will be discussed in the next section.

4 Diphone preprocessing

In order to extract rules from a diphone database, some preprocessing is required to get rid of the data-intrinsic noise. This 'noise', viz. the effect of systematic and non-systematic errors, stems from various origins:

1. only one realisation of each diphone is considered
2. the diphone segmentation may be suboptimal
3. the LPC-paradigm as a model is sometimes inadequate (e.g. in the case of nasals and plosives)
4. usually, only 10 LPC-parameters are used to describe the spectral envelope
5. an adapted version of the LPC-method (Split-Levinson) is applied in order to obtain continuous parameter tracks: bandwidths are more difficult to obtain
6. the analysis windows have been positioned pitch-asynchronously
7. the LPC-optimization algorithm introduces truncation errors
8. the de-emphasis is chosen suboptimally

The first and second point are methodological ones. The construction of a diphone set involves recording, (automatic) diphone segmentation, individual diphone corrections and a proper diphone duration definition. The third point deals with the LPC-method itself, as opposed to e.g. short-window Fourier analyses. The other items refer to the various settings within the LPC-method, inherent to the software available at IPO.

In order to reduce the data noise, one may stylize the data such that inessential information is discarded. This has been our approach in ALLODIF. Another method is, to use several instances of each diphone in parallel, and to look for common features in such 'diphone ensembles'. Such ensembles, however, are not available for one Dutch speaker. We discuss the stylization method in the next subsection.

4.1 Diphone stylization

In order to obtain information from diphones a diphone stylization has been performed. The principle is shown in figure 1. The dotted lines indicate original parameter tracks, full lines represent the stylized version. Time is plotted along the horizontal axis. The stylization results in parameter tracks such that all spectral parameters F_i and B_i, $(i = 1, \ldots, 5)$ fulfill the following properties:

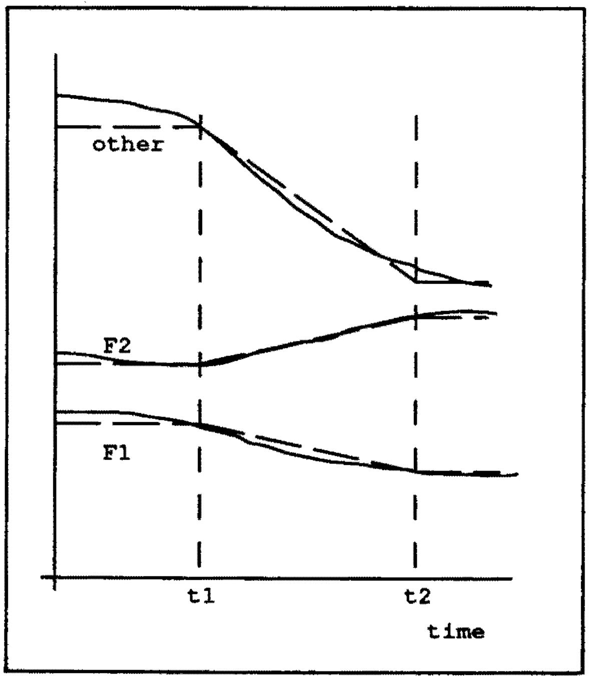

Figure 1: The stylization of a diphone. The full lines represent the original tracks; the dotted lines indicate the stylized version.

- All parameters are either constant or change linearly over time,
- Periods during which all parameters are constant may occur diphone-initially and diphone-finally,
- In between, they all move linearly over time.

The first point forbids any polynomial dependence of time with degree higher than one. Although higher order approximations may be useful for narrow modelling of the data, there is no indication that quadratic, cubic or exponential approximations are required perceptually.
The second point refers to the possibility that any of the constant parts in the stylization is absent. This may happen if the diphone contains mainly a dynamic part. In the third point, 'change linearly' means that a parameter p can be described by

$$p = \alpha + \beta t$$

for some fixed α and β.

The stylization was performed by a minimization algorithm, which allows a 'tuning' in different ways. For example, the claim of simultaneity between all parameters can be dropped (such that for example F_1 moves independent of other formant parameters), and different weighting of the parameters can be applied. The ultimate choice for the algorithm settings in the present software version has been motivated by informal listening tests (see below).

These settings were used to construct the stylized Zelle diphone set and the stylized Bloemendal diphone set, both in the LPC-10, 10 kHz-version. The algorithm was optimized to a point after which it appeared difficult to improve the perceptual quality of some set of stylized diphones without seriously damaging the quality of other diphones.

The stylization allowed us to code each diphone with a set of well-defined parameters. These parameters are:

- the five formant frequencies, five bandwidths, energy and voicing of the first steady part (the initial target);
- the twelve same parameters of the final target;
- the moments t_1 and t_2 representing begin and end of the spectral intertarget transition phase;
- moments to indicate the transition phase for the energy contour;
- the moment of the change of the voicing parameter, and
- the duration of the entire diphone.

This yields a total of 30 parameters, so-called anchor points: a reduction of about a factor five compared to the 'original' diphones. All the anchor points can be used as input for the optimization algorithm that looks for the relation between those anchor points and the context, more specifically: the relation between the phonetic anchor points and the phonological feature values of the context. We come to this in the next section.
However, the price we pay for this possibility is that we extract rules from a stylized diphone set which may be slightly affected by the stylization. So, a priori, speech synthesis based upon rules that are extracted from diphones will provide speech of at most the same quality. At the beginning of the project it was not quite clear to which extent this problem would become a serious one. There was no reason to be pessimistic beforehand, as allophone synthesis allows much more parametric freedom than diphone synthesis and the rules can be improved by careful redefinitions of the anchor points. In principle, rules of the four types we discussed before can provide synthesis of high-quality speech. Therefore, the often-heard argument that speech based on rules derived from diphones cannot be better than the diphone speech is not true.

The stylization algorithm is rather straightforward from a mathematical point of view. Its implementation, based upon a lot of bookkeeping details, took a few months. The present implementation is suitable for any parameter input as long as these parameters specify moments of specific spectral and temporal details. It is required that the input data contain continuous parameter tracks (see also section 4.3).

4.2 A preliminary perception experiment

In order to find optimal stylization settings, the resulting stylized utterances have informally been judged by a panel. Intelligibility tests of *VCV* and *VCCV* utterances were carried out by eight listeners. Four subjects were presented with 80 of both the 'original' (diphone) version and the stylized utterances. Another four subjects only heard 80 stimuli of the stylized versions. Both groups had to write down what they heard; the first group was also asked to indicate whether they heard

- no difference,
- a difference but only at the phonetic level (free variation), or
- a difference on phonological level (contrast).

From these tests, the following perceptual results were obtained. In about one third of all stylized cases, no difference was heard analytically between original and stylization. In another thirty percent, a difference was perceived in the analytic listening mode. In the remaining cases, about one half showed a better intelligibility, the other half performed worse. The conclusion could be drawn that better stylization techniques could probably be obtained only if the stylization method is made diphone-specific. In other words, the present stylization is among the best that are valid for all diphones simultaneously.

The above test results are in reasonable agreement with the findings obtained from extensive evaluation results Van Bezooijen (1990). She finds that both the stylized diphone sets performed about 10 % worse than the original sets in a well-controlled open reponse segment identification task. The subjects were presented with non-lexical, phonotactic stimuli of the form *CVC*, *CVVC*, *VCV* and *VCCV*. For further details we refer to Van Bezooijen (1990).

4.3 The choice of the diphone set

In our project, we mainly examined the diphone set spoken by H. Zelle. In the project BLOEMDIF (by R. Drullman), a diphone inventory was constructed of speech by Ph. Bloemendal. Bloemendal was chosen to be a 'norm speaker' in the ASSP-program. This inventory consists of non-reduced ('full') and reduced diphones, and was originally recorded with a sample frequency of 20 kHz and analysed with LPC-30. Since this set provides high-quality speech, it has been attempted to apply this Bloemendal-set for stylization purposes.

For two reasons, however, the LPC-30 version of Bloemendal was not appropriate. Firstly, we observed that in case of high-order LPC, the numerical limits of some essential portions of the LVS software were met. Therefore, the Bloemendal-set was for our purposes re-analysed with LPC-10 (with a

Split-Levinson algorithm), which resulted in a deterioration of the resulting speech naturalness.
Secondly, as we already observed, in order to find systematic details in parameter values, there is a requirement for continuity of formant tracks. All poles found in the LPC-analysis must be 'relatable' to a formant position. The formant frequencies and bandwidths obtained with LPC-10 at 10 kHz sample frequency obey this condition; the 15 complex poles obtained from LPC-30 at 20 kHz, however, do not: one specific pole may now gain and loose its relevance for describing spectral details. To go into details would be beyond the scope of this report[3]. It is certainly of interest, to reconsider the usefulness of the LPC-30 version of the Bloemendal diphone set when the formant-tracking problem is technically solved.

[3]As a spin-off of the project, the parameter conversion routines of LVS have been rewritten by using double precision fast NAGLIB routines. Furthermore, an algorithm has been designed to cope with the spectral difference where the spectra are specified by pole positions in the complex plane. This algorithm has not fully been tested, however, as the decision was made (in project BLOEMDIF) to go further with LPC-10 at 10 kHz.

5 Automatic rule extraction

Once we have a stylized diphone set, we are prepared to look for relations between the anchor points in the diphones resulting from the stylization, and the segmental context. In this section we will discuss a method to explicitly find these relations. The method that we will propose will be referred to as the 'IPO-KUN-method' for simplicity. Other methods will be mentioned and discussed in section 5.2 and in appendix C.

All rules we encountered until now have the form

$$\text{if } C \text{ then } A$$

where C and A denote the context specification and the action, respectively. The context specification yields a unique condition for application of the rule. Our method is based on the idea that rules of the form 'if C then A' can be translated to a *condition-free* rule consisting of an action 'A_2' only. In the latter case, the action A_2 is more elaborate than the action A. The action A_2 in the new rule combines the condition C *and* the action A in the old rule. Conversely, a condition-free rule with complicated action A_2 can be translated into a rule of the form 'if C then A'. Since this latter rule has a format close to the format of the the Nijmegen rules, it is called 'phonological'. The packed version A_2 will be called 'numerical'.

The IPO-KUN-method consists of two steps:

1. to bring about a translation from phonological rules to numerical rules and vice versa
2. the algorithmic search for numerical rules in a database

The line of thought is as follows. Assume we want to govern by rule some phonetic parameter, say F_1. By means of the algorithm in the second step, we look for numerical F_1-rules that are valid on the whole database or an appropriate subbase that is as large as possible. Once such rule is found, we use the first step to translate the numerical F_1-rule into a phonological F_1-rule, and we put the discovered rule into a rule set. And reversely, once we have found a phonological rule, we can use both steps to transform it into a packed numerical rule.

In section 5.1, we will deal with the first step. We here consider a simple example; for more details we refer to ten Bosch (1990). The second step will be discussed in section 5.2.

5.1 The conversion from phonological rules to numerical rules and vice versa.

In this section, we give an example of the conversion between phonological and numerical rules. In this example, we assume to have a speech database at hand. In this database with labeled speech, the allophone [a] is assumed to occur sufficiently many times to allow its spectral state to be properly defined. The first formant F_1 of [a] is on average equal to 700 (Hz), except for [a] in unstressed syllables where the average F_1 equals 650. We say that the *parameter* F_1 depends on the *function* [stress]. For simplicity, we here assume that this function is binary-valued only. Other functions are e.g. [height], [front], [round], [plosive], [nasal], [voice], [fric], etc. Other parameters are e.g. the formant frequencies F_i, bandwidths B_j, the energy, LPC-parameters, etc. Functions and parameters are phonologically and phonetically inspired, respectively[4].

The corresponding rule might read:

$$\begin{array}{ll} \text{if focus} = \text{[a] and syllable is unstressed then} & F_1 := 650 \\ \text{if focus} = \text{[a] and syllable is stressed then} & F_1 := 700 \end{array}$$

or

$$\text{if focus} = \text{[a] then} \begin{cases} F_1 := 650 \\ F_1 := F_1 + 50 \text{ if [stress](syll)} \end{cases}$$

Here '[stress]' is formulated as a predicate with 'syll' as an argument, defined by

$$\text{[stress](syll)} = \begin{cases} 1 \text{ if the syllable syll is stressed} \\ 0 \text{ if the syllable syll is unstressed} \end{cases}$$

In this example, [stress] is a binary-valued function. If the two values are 0 and 1, we call it a *boolean*. Other functions (e.g. [height]) may attain more different values.

Also for the focus, we can introduce the predicate 'F_{a}' with argument 'focus' by defining

$$F_{\text{a}}(\text{focus}) = \begin{cases} 1 = \mathit{true} \text{ if 'focus' equals [a]} \\ 0 = \mathit{false} \text{ otherwise} \end{cases}$$

Accordingly, the above rule has a 'numerical' variant:

If $F_{\text{a}}(\text{focus})$ then $F_1 := 650 + 50\ \text{[stress](syll)}$

[4]In appendix B, we show the phonological functions used in the Nijmegen allophone synthesis.

or, explicitly,

$$\begin{aligned} F_1 &:= (650 + 50[\text{stress}](\text{syll})) \cdot F_{\text{a}}(\text{focus}) \\ &= 650 \cdot F_{\text{a}}(\text{focus}) + \\ &\quad 50 \cdot [\text{stress}](\text{syll}) \cdot F_{\text{a}}(\text{focus}) \end{aligned} \qquad (1)$$

We see that the final, 'numerical' rule 1 corresponds to the initial rule 'in words' we started with. We found a translation from a phonological rule to a numerical rule. This example provides a numerical rule with two functions, two terms and degree two. More difficult rules can be formulated by using more functions, more terms and higher degrees, but they involve exactly the same kind of logic. It can easily be seen that the compound factor in the above rule 1 ($[\text{stress}](\text{syll}) \cdot F_{\text{a}}(\text{focus})$) corresponds to the 'if-nesting' in the phonological rule we started with ('if focus = [a] and (if) the syllable is (un)stressed, then ...'). Consequently, the complexity of the terms in the numerical rule (A_2) determines the complexity of the context specification (C) and action specification (A) in the phonological version.

It will be evident that many phonological rules can be translated into their numerical variants. There is, however, a trade-off between context specification and action specification in the rules. In the rule we started with:

if focus = [a] and syllable is unstressed then	$F_1 := 650$
if focus = [a] and syllable is stressed then	$F_1 := 700$

the context specification is rather elaborate ('focus must be [a]', 'syllable must be stressed or unstressed') while the action is very simple: $F_1 := 650$ or $F_1 := 700$. In the last rule:

$$F_1 = 650 \cdot F_{\text{a}}(\text{focus}) + 50 \cdot [\text{stress}](\text{syll}) \cdot F_{\text{a}}(\text{focus})$$

we do not have *any* context specification, but the action is rather elaborate. Schematically, there is a balance between actions and contexts:

phonological rule		numerical variant
context } action	⇔	action

We have seen that the degree of the terms in the numerical rule determines the complexity of the action that can be represented by it. By the translation between numerical and phonological rules, it follows that the maximal degree that we allow in the numerical rule determines the complexity of the context of the corresponding phonological rule. If the context of a particular phonological rule is very narrowly specified, e.g. by '⋆ a', where '⋆' represents the focus and 'a' denotes the right-hand side phoneme, then we will need a high degree in the numerical variant in order *to be able* to find the numerical variant. The more specified a context of a phonological rule is, the higher will be the required degree in the equivalent numerical rule. Therefore, since the degree of terms in the action of numerical rules directly corresponds to the number of nested 'if's' in the phonological rule, it is relevant to estimate the maximal depth of 'if-nestings' in the Nijmegen set. A study of the rules (the version of October 1990) led to an *average* depth slightly larger than 3. If we confine the discussion to single phonemes, we already observe a great variation in required number of if-nestings. That number varies from one (in the case of the segment 'r') to five (in the case of the segment 't'). All other segments fall between these extremes. In appendix B, we consider the Nijmegen symbol set and feature set in more detail.

In this section, we have seen the relation between phonological rules and numerical rules. Using this relation, we discussed which type of phonological rules are easy to find, and which type is more difficult to discover. On the strength of this relation, we can reduce the problem to the search for numerical rules. In the next section, we discuss how to find such 'numerical' rules in a semi-automatic way.

5.2 Algorithmic search for numerical rules on a database

We observed that the IPO-KUN-method consists of two steps:

1. to bring about a translation from phonological rules to numerical rules and vice versa
2. the algorithmic search for numerical rules in a database

In this section we discuss the second step.

The method that we developed in ALLODIF to solve the rule-extraction problem is based upon ideas developed in ten Bosch (1989, 1990). It is based on the interpretation of the solution of a minimization problem, and it makes use of a matrix formalism. Our method is not the only possible one to extract numerical information from databases. Other methods include: clustering analysis (CA), the CART-method, discriminant analysis (DA), the covariance method (COV), and neural network approaches (NN). Our method is a kind of combination of the CART-method and discriminant-analysis. The five methods CA, CART, DA, COV, and the IPO-KUN-method are in contrast with the NN-methods as the former allow the derivation of *rules*, whereas the NN-approaches still lead to 'black boxes': i.e. networks with weights from which it is difficult to extract information how they work. The IPO-KUN-method and the COV-method seem to be the most appropriate

Table 1: A representation of a speech database

Functions				Parameters		
F_a	...	[stress]	...	...	F_1	...
0	...	0	...	...	510	...
0	...	1	...	...	350	...
1	...	1	...	...	700	...
1	...	1	...	...	700	...
1	...	0	...	...	650	...
⋮						
0	...	0	...	...	405	...

to effectively derive rules. In appendix C, we mention the specific, rather technical, differences between these methods.

The second step, viz. the search of numerical rules, consists of three substeps:

- (re)arrangement of the data
- algorithmic search of an adequate minimizing expression
- interpretation of the obtained minimal expression

In our example, we use the case of the [a]-rule that we discussed earlier:

if focus = [a] and syllable is unstressed then $F_1 := 650$
if focus = [a] and syllable is stressed then $F_1 := 700$

that was translated into

$F_1 = 650 \cdot F_a(\text{focus}) + 50 \cdot [\text{stress}](\text{syll}) \cdot F_a(\text{focus})$

First substep: rearrangement
The first step involves a representation of the speech database in a tractable way. One possible representation is shown in table 1. Such a matrix representation is very basic and often used in other methods as well (e.g. in the method by Van Santen & Olive). In the matrix presented, we read the values of the F_1 in the F_1-column at the right side. The left-hand side consists of the values of phonological features and other relevant context features. We here assume that the data concerning [a] are accurately described by the [a]-rule.

Second substep: algorithmic search
The second step deals with minimization of some numerical expression. The above F_1-rule can be found automatically from table 1 by considering all lines in the table that contain a '1' in the function column F_a. We then construct a matrix A which is derived from the left-hand side of table 1. After that,

the minimization of a vector expression $||Ax - b||$ is considered, in which b is a known parameter vector containing the parameters in the F_1-column in the right-hand side, and x is the unknown weighting vector consisting of the weighting coefficients (Golub & Van Loan, 1983). The problem now is: how do we construct matrix A?

We now come to an essential observation: *If* A contains at least the three columns F_a, [stress] and their column product, *then* the expression $||Ax - b||$ can be minimized towards zero. The vector x will then denote the weighting and we find the numerical rule, i.e. an approximation of b, by expanding Ax. The choice of the matrix A determines the vector x and, accordingly, the approximation of the parameter vector b. The search for a numerical rule Ax is now reduced to the search for the optimal matrix A. In practice, data are distorted by noise and the resulting rule will only approximate the data.

If Ax is expanded, it will have the following polynomial form (p denoting the parameter from the vector b, [fun] a function):

$$p := \overbrace{a_0}^{\text{constant}} + \overbrace{a_i[\text{fun}]_i + \ldots}^{\text{linear terms}} + \overbrace{a_{ij}[\text{fun}]_i\ [\text{fun}]_j + \ldots}^{\text{quadratic terms}} + \ldots \tag{2}$$

Third substep: interpretation
The algorithm searches for the matrix A and the vector x such that Ax optimally approximates b. The constant term in the right-hand side in equation 2 represents the table default value of the parameter p. The linear terms corresponds to the correction of the table value by one if-statement, the second order terms correspond to the correction by two if-statements, et cetera.

In the present algorithm we impose two restrictions on the structure of the matrix A, one dealing with the number of columns and the other dealing with the degree of the columns. For that purpose we define two integers M and E. The number M denotes the maximal number of columns of A; as we have seen, this corresponds to the *length* of the corresponding phonological rule. The second number E denotes the *maximal degree* of columns of A, more precisely, of the entries in those columns. By restricting M and E a priori we squeeze the search space to feasible size. The number of columns in the input data matrix is not restricted.

We present an example. Suppose the datamatrix has the form

	1	2	2	1
	-3	-4	1.2	3
	0	0	1	1
column	c_1	c_2	c_3	c_4

The matrix A can be constructed by applying the algebraic operations '(c_1c_2, c_3^2, c_4)' which yield

$$A = \begin{pmatrix} 2 & 4 & 1 \\ 12 & 1.44 & 3 \\ 0 & 1 & 1 \end{pmatrix}$$

In this example, $M = 3$ and $E = 2$. The restriction on the matrix A by M and E is important for two reasons. One reason deals with the CPU-time. The required CPU-time is more than polynomial in both M and E; from this point of view the algorithm is far from optimal. For theoretical reasons, however, it is not easy to reduce the search space such that identical results are produced with high probability in polynomial time (Lenstra, pers. comm.). The second reason to restrict the values of M and E deals with the interpretability of the minimization results. The solution vector x and the approximation Ax will have phonetic 'sense' only if the number of columns is limited, and if the rule is not too complicated. For the interpretation of the expression Ax it is very useful to be able to factorize the involved polynomial in order to obtain a neat phonological rule interpretation, as every factor corresponds to a new 'if-level'. For the description of the data itself such a factorization is not required. Unfactorized polynomials, however, may yield unnecessarily difficult phonological rules which do not provide any insight.

As a consequence, the choices for E and M are essential for the ability to find appropriate rules. If these parameters are set too small, the search space is too small and we will probably not find a rule at all. If they are too large, however, the algorithmic search of an optimal rule may take a very large amount of time. Therefore, a proper determination of their values is crucial. We mention two methods for that.
Firstly, we can and probably *have to* incorporate intuitive phonetic knowledge to simplify our task and to reduce the dimension and size of the search space. The search for A can considerably be shortened if factor interaction is globally known beforehand. For example, the duration of vowel phonemes is known to depend on phonemic identity, accentuation, speaking rate and voicing of the following consonant. A preselection of these four factors reduces the search space as far as possible.
Secondly, we can derive some estimates from the Nijmegen rule set. Such estimations depend on the context specifications in the Nijmegen rules, but also on the particular set of features in the left-hand side of table 1. In general, M can be set to 5; this corresponds to a parameter dependence of maximally five terms in the expansion of the rule Ax. Further, the maximal degree E can be limited to 3. This is a safe upperbound, if the focus and context specification is broad. This can be argued as follows.
The powers of the columns of A determine the complexity of the action in the numerical rule. This action can be translated into both a context specification and the action of a phonological rule. Therefore, the degree of columns of the matrix A has to be distributed over the focus and context specification. In order to be able to find complex actions, we must simplify the context specification as much as possible. If the focus can be specified without any

Table 2: The l-rule in the Nijmegen rule set.

```
1 ->>      1 FORM1 -:= 70 / --- {u/i/y/@}
1 ->>      1 FORM2 -:= 200 / --- [+voc,+achter]
1 ->>      4 SET F3=F2
             FORM3 +:= 200
             BANDBR3 := 200
             FORM4 := 2500 / --- [+voc,+rond]
1 ->>      2 BANDBR1 := 80
             BANDBR2 := 80 / --- {u/y}
1 ->>      2 BANDBR3 := 400
             BANDBR4 := 150 / --- {O/o}

1 ->>      6 FORM1 := 400
             FORM2 := 810
             FORM3 := 2750
             BANDBR3 := 150
             FORM4 := 3375
             BANDBR4 := 100 / [+voc] ---
```

feature specification, $E = 3$ can be used fully in the context section, which allows an if-nesting of depth ≤ 3. An example of such a feature-less context specification is provided in the next subsection where we deal with the l-rule. In the discussion of the Nijmegen rule set we observed that some rules have a context specification of one segment only. As many segments need four features for unambiguous determination (see appendix B), this kind of rules can probably not be found by the algorithm without manual preselection of the context. In the present Nijmegen rule set, rules with a context of precisely one phoneme are indispensable to obtain its present specific input-output characteristic (Loman, pers. comm.). From the point of elegancy and optimality of rule improvement, the number of these one-phoneme-context rules should be maximally reduced.

5.3 Example: the l-rule

As an example, we consider the behaviour of the prevocalic /l/. In the Nijmegen rule set, the /l/ is dealt with by the rules shown in table 2 (the 'l-rule').

In the upper ten lines, the /l/ is adapted according to the next vowel. After F_1 and F_2 are dealt with, the third formant F_3 is set to $F_2 + 200$ with a band width of 200 before round vowels. It is the assignment of F_3 we are now

Table 3: The datamatrix derived from the Zelle diphone set for the l-rule.

```
featurefile read:  [tenbosch.rules]foneem_feature.dat
indexed file read:  data1disk:[difonen]zelle_stand.ind
          vow   low   fr    voi   len   rnd   f1     f2     f3
L1A1      0.    3.    1.    0.    1.    0.    352.   1263.  2871.
L1AA1     0.    3.    1.    0.    2.    0.    357.   1399.  2820.
L1C1      0.    2.    1.    0.    1.    0.    258.   1453.  2645.
L1CC1     0.    2.    1.    0.    1.    0.    260.   1405.  2691.
L1E1      0.    2.    2.    0.    1.    0.    334.   1528.  2637.
L1EE1     0.    2.    2.    0.    2.    0.    283.   1582.  2750.
L1I1      0.    1.    2.    0.    1.    0.    265.   1560.  2811.
L1II1     0.    1.    2.    0.    2.    0.    269.   1719.  2609.
L1O1      0.    2.    0.    0.    1.    1.    299.   1098.  2857.
L1OE1     0.    1.    1.    0.    2.    1.    257.   1506.  2588.
L1OO1     0.    2.    0.    0.    2.    1.    271.   1453.  2634.
L1U1      0.    1.    0.    0.    1.    1.    239.   1477.  2240.
L1Y1      0.    1.    1.    0.    1.    1.    181.   1584.  2488.
```

interested in.

The l-rule, found by H. Loman in 1989, resulted from careful listening and inspection of spectrograms. We have attempted to find a similar rule in the diphone data. Therefore, first a datamatrix was constructed (table 3).

In the first column, all 13 diphones are shown in which /l/ is in initial position with a monophthong in final position. The last three columns contain the values of F_1, F_2 and F_3 in the initial frame, respectively. The mid section in the table contains a matrix with 6 columns and 13 rows. These 6 columns represent the values of the features [vowel], [low], [front], [voice], [length] and [round/lab] for the post-/l/ vowel phoneme, respectively. Columns that contain identical entries only are set to zero.

We observe that, in no case, the difference between third and second formant frequency is less than 700 Hz. Therefore, it is impossible to find a rule such as

```
...
  1 ->>      4 SET F3=F2
               FORM3 +:= 200
               BANDBR3 := 200
               FORM4 := 2500 / --- [+voc,+rond]
...
```

Table 4: The l-rules for the first three formant frequencies, derived from the Zelle diphone set.

parameter	constant term table value	correction	error (st.dev. in Hz)
F_1	185	53·[low]	(30)
	228	14·[low]^2	(29)
F_2	1686	-126·[low]	(132)
	1324	130·[front]	(126)
	1362	60·[front]^2	(125)
	1346	76·[front][length]	(123)
	1373	37·[front]^2[length]	(122)
F_3	2398	150·[low]	(139)

By the rule-extracting algorithm, the behaviour of F_1, F_2 and F_3 can be considered in terms of the features [low], [front], [length] and [round] of the vowel. The results are shown in table 4.

In this table, each line represents a rule. The corresponding error, measured in terms of standard deviations, is shown in the last column. In the second column, the default values of F_1, F_2 and F_3 are shown. These values are the most appropriate to be put in a table in which the spectral states of context-free allophones are defined. One observes that there might be no 'unique' default value: in case of the second formant, the table value ranges from 1324 to 1686. This will become clear when we take into consideration the fact that default values are 'default' only with respect to all possible later modifications, rather than representing an 'average value'. For example, if F_1 were always be equal to 500 Hz except in stressed syllables where $F_1 = 600$, and 30 percent of all syllables are stressed, then the default value for F_1 had better be equal to 500 rather than to the average

$$\frac{7}{10}500 + \frac{3}{10}600 = 530$$

In the latter case, the rules become unnecessarily complicated.

In the third column, the first context-dependent modification of the default value is shown. The expressions [low], [front] and [length] represent values to be taken from the datamatrix used as input for the algorithm. For example, in the F_2-rule on the fourth line of table 4:

$$F_2 = 1324 + 130 \cdot [\text{front}]$$

the parameter [front] attains the value of the feature 'front' of the post-focus vowel, presented in table 3. The prominent presence of [low] and [front] in all the lines of table 4 can be understood as these features correlate with the first two formant frequency values.
In case of the third formant F_3, the parameter [round] is expected to explain a great deal of the modification, since this parameter turns out to be very useful in the Nijmegen set. This is, however, not found in the diphone set. The parameter [low] turns out to explain more data than the parameter [round], in which case the approximation is

$$F_3 = 2729 - 167 \cdot [\text{round}]$$

with an error of 156 Hz.

We conclude that the Nijmegen l-rule, which states that F_3 approximates F_2 up to 200 Hz before round vowels, cannot be traced back in this particular diphone set (the Zelle set). From an optimization point of view, the feature 'round', which is chosen in the Nijmegen set in the modification rule as a prime context specifier, cannot be found either. The data in the diphone set, however, can be described by other rules. This means that, in general, the Nijmegen-rule set cannot be improved by the present rule extraction approach. Our approach needs its own specific rule set, based on the output of the rule extraction algorithm presented here. These rules may yield acceptable output without being interpretable in a phonetic way. We return to this point in section 8.

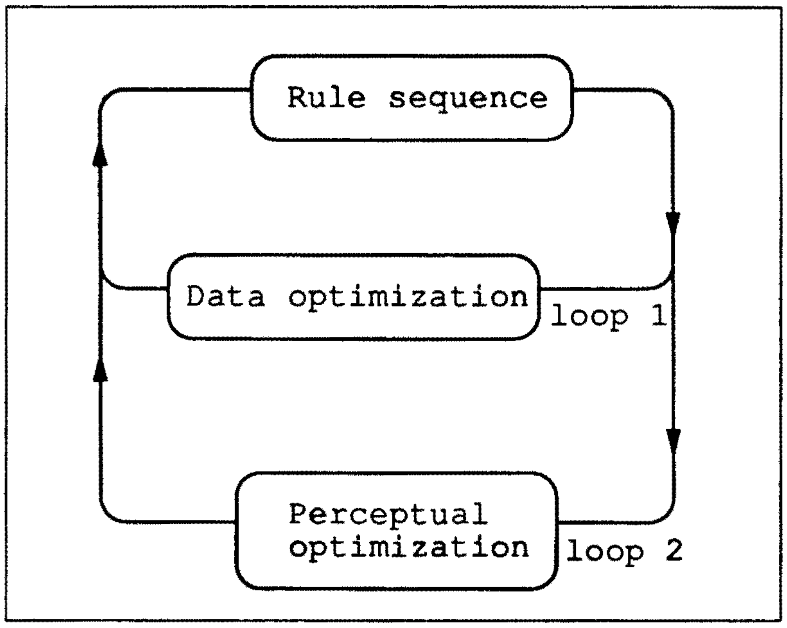

Figure 2: The proposed ideal loop for optimizing rule sets. The data error as well as the perceptual error are dealt with in the loop.

6 Perceptual relevance of rules

Until now, we discussed the possibility of extracting rules in some format from a datamatrix. The current optimization only deals with the interpretation of the data as found in the database. There is no perceptual feedback. Data error minimization procedures must not be expected to improve the *perceptual quality* of the resulting allophone speech, since the relations between speech quality and speech parameters are too complex. A perceptually-inspired distance function based on dynamically changing acoustic parameters has not been found yet. It is my belief that such a function does not exist at all.
An alternative is, to improve the quality of the rule output by introducing an explicit perceptual evaluation phase in the rule optimization scheme. This option leads to a optimization scheme shown in figure 2. In order to produce optimal rules, first a good 'first guess' for the rule set is produced on the basis of the goodness of fit of speech parameter data. After having produced such a set, it is further improved by perceptual 'tuning' in the perceptual evaluation phase.

The numerical algorithm is here assumed to produce a first 'good guess', based upon speech data, for an allophone rule set. Such a good guess must be improved by using other, perceptual criteria. Perceptual rule improvement, however, requires a fast rule interpreter which allows the user to interactively modify rules in text-format in real time. Due to lack of time, such an interpreter has not been developed yet.

As a spin-off of the project, however, a rule-formalism and a *preliminary* interpreter have been designed and implemented (appendix D). This formalism and algorithm allow a real-time rule editing with near-real-time speech response. The formalism is very similar to the formalism applied by Klaus Wothke of IBM Scientific Center in Heidelberg (Wothke, 1990), and is characterized by a very strict rule syntax and semantics definition. Only in such strictly defined environments can a rule interpreter be subject to more or less automatic optimization. We now have a dilemma: on the one hand, the Nijmegen set certainly meets a 'human readability' condition, but on the other hand, its structure is not likely to be optimal from the optimization point of view. The readability is based on the use of phonetic features that yield insight in the systematic pattern in the speech production process, and on the use of the standard phonological format. The second point (non-optimality of the Nijmegen set) has to do with the rule structure itself as well as the structure of the rule *sequence*. This latter structure will be discussed in the next section.

7 Application in rule sets

In the preceding sections, we examined a method to extract rules from a speech database. These rules have to be of special form, as we have seen above. This form results from the rule derivation by means of matrices. Each approximation Ax to b yields a rule for the phonetic parameter corresponding to b.

In this section, we will consider ***rule sets*** rather than separate rules. Since the term 'rule set' suggests that rules are to be applied without any specified ordering, the term 'rule sequence' is to be preferred. In the sequel we will therefore interpret 'set' as a 'sequence'.

A rule will be denoted by R_i. We have seen that the rule R_i is decomposable into one parameter setting and a sequence of compound if-statements. A rule set, such as the Nijmegen set, can be represented by ordered sequence of rules

$$(R_1, R_2, \ldots, R_k)$$

where R_{i+1} appears 'later' than R_i.

Our goal in this section will be to show that the proper construction of such a sequence is not trivial. In order to show some of the peculiarities involved, we consider a simple example.
Suppose we have a rule sequence for generating allophone speech $(R_1, R_2, \ldots, R_k)$. This rule sequence will somewhere have to deal with the focus /p/. We now concentrate on the /p/. The phoneme /p/ is phonetically determined by [labial], [plosive], and [voiceless]; accordingly, /p/ will result as outcome of all rules dealing with unvoiced phonemes, plosives, and labials. All these rules have a specific focus-context-specification, of which the intersection is /p/.
For simplicity, we now assume to face two possibilities. Either the /p/ results correctly, i.e. the rule sequence yields an intelligible /p/ in every context, or /p/ results incorrectly in some context. We are interested in the second case. If the /p/ results incorrectly and other voiceless phonemes, labials and plosives result correctly, we have the following problem: How to modify the voiceless-rules, the plosive-rules and/or the labial-rules in such a way that

- /p/ is improved
- all voiceless phonemes other than /p/ are not affected by the alternative voiceless-rule
- all plosives other than /p/ are not affected by the alternative plosive-rule
- all labials other than /p/ are not affected by the alternative labial-rule

One can imagine that there might exist a large factor interaction in such a rule sequence: Improvements in earlier rules have their consequence in later rules. In this example, the improvement of the voiceless-rules, the plosive-rules as well as the labial-rules may turn out to be too difficult, due to this interaction. Instead of improving these three rules, we may be be forced to adopt one extra /p/-rule at the end of the rule sequence that specifically corrects the results of the three former rules. In other words, a later correction rule is introduced with a context that is a subcontext of earlier rules, depending on how the rule set precisely acts on the input phoneme string. (A description with a precise analysis of these facts would go beyond the scope of this report. For details we refer to Ten Bosch (1991).)

If we analyse the above problem, we come to the following conclusion. The rule-improvement is hampered by the presence of many rules of which the context specifications intersect. This phenomenon led Van Santen to his rejection of rule sets to describe phoneme duration phenomena. I am a bit more optimistic. In the following, I will attempt to show the optimal structure of a rule set. In the sequel, it is assumed explicitly that the focus and domain specifications are defined on one level, i.e. that no suprasegmental information occurs in the segment input string.

We assume that the input string is a finite string of symbols. The symbols are taken from a universe symbol set. For simplicity, we assume this universe set to be all one-place characters that 1-1 correspond to Dutch phonemes (via a table). So 'a' represents /a/, 'A' /α/, etc. Therefore, 'symbols' is in this context synonymous for 'phonemes'.

The focus and context specifications are exclusively based on the set notion. For example, [plos] denotes the set of plosives. [plos] is defined by enumeration (i.e. '[plos] contains /p/, /b/, /t/, /d/, ...') or by implicit definition by means of a feature ('[plos] contains everything that has a plosive-value equal to 1, and nothing else'). These definitions are equivalent and are both used in the Nijmegen rule set. A focus specification might read as follows

- focus = [plos], or
- focus = 'p'

and, similarly, a context definition may look like this

- [plos] [vowel] focus {p/t/k}, or
- focus [vowel]

denoting a string containing a plosive, a vowel, the focus, and a /p/, /t/ or /k/, and a string containing a focus and a vowel, respectively. In general, the context definition[5] is

[5]We here disregard the possibility of the specification [nonseg]0 in the context definition.

$$\ldots S_{-2} S_{-1} \text{ focus } S_1 S_2 \ldots \tag{3}$$

in which S_i denote rule-specific subsets of the universe symbol set. The rule fits if the focus specification holds as well as the context specification holds.

It is easy to see that

$$S_{-2} S_{-1} \text{ focus } S_1 \tag{4}$$

specifies a broader context[6] than e.g.

$$S_{-2} S_{-1} \text{ focus } S_1 S_2 \tag{5}$$

or

$$S_{-2} S_{-1} \text{ focus } \{e\} \tag{6}$$

with e one element from S_1. However, we remark that context 5 is in general incomparable[7] to context 6, if S_1 contains contains other elements than e, if S_2 is a *proper* subset of the universe symbol set, and if the actions of rule 5 and rule 6 deal with the same acoustic parameter. We assume here that this is the case. Due to the context incomparability, the introduction of rules with contexts 5 and 6 leads to the optimization problem described above.

One way to solve this problem consists in a redefinition of the rules with the following consecutive contexts

$$S_{-2} S_{-1} \text{ focus } \{S_1 - e\} \tag{7}$$

$$S_{-2} S_{-1} \text{ focus } \{S_1 - e\} S_2 \tag{8}$$

$$S_{-2} S_{-1} \text{ focus } \{e\} \tag{9}$$

where $\{S_1 - e\}$ denotes the set S_1 minus the element e. According to the remark above after context 6, this reduced set is not empty. In figure 3, the situation is depicted in the case of the contexts 4, 5 and 6 at the left side, and contexts 7, 8 and 9 at the right side.

[6] A context C_1 is broader than context C_2 if all input strings fitting C_2 also fit C_1.

[7] This here means that there exist input strings that fit one context but not the other, and vice versa.

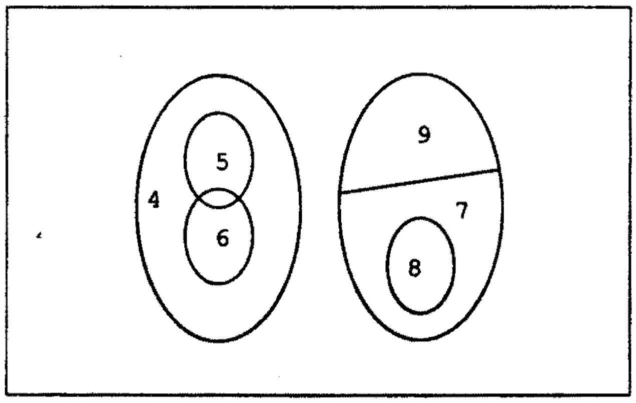

Figure 3: Venn diagrams of various contexts. For an explanation see the text.

Now, we define a tree structure on the rule set, as follows. Rule R_1 is said to be mother of rule R_2 if the context of R_1 is broader than the context of R_2; R_2 is daughter of R_1. One observes that this definition does not yield a correspondence with daily life, as for example a daughter's daughter is not usually a daughter. *Comparable* contexts define mother and daughter. Every rule has a mother, since the *default table* can be interpreted as a collection of the broadest rules which are all sisters.

In the above example, the rule 4 with identical foci and context, is the mother of the two sisters 5 and 6. These sisters, however, are incomparable and not disjoint[8]. Context 7 is the mother of context 8, but is now the sister of context 9. These sisters are disjoint. We can now formulate the conditions for a rule sequence to be optimal: *all sisters must be disjoint.* In ten Bosch (1991) it is shown how an arbitrary rule set can be transformed to satisfy this condition.

In the project ALLODIF, we have investigated the optimal theoretic structure of a rule sequence aiming at the transformation from linear symbol strings to actions on phonetic parameters (ten Bosch, 1991). Unfortunately, time was lacking to apply the results on the Nijmegen rule set and to actually rearrange the rules. If the work in the project ALSYS is to be continued, I would propose first to study the rule set structure in more detail, by invoking a simple set inclusion test module with the Nijmegen interpreter as 'front-end' and collecting statistical and structural data on a large number of balanced or random symbol sequences as input sentences.

[8]Two rules are disjoint if no input strings exist that fit both rules.

8 Discussion

In retrospect, in the project several questions have been studied which deal with the rule-extraction-problem. Software for diphone stylization (with several options) STYLIZE has been developed. Algorithms for the construction of data matrices, for the search for a best fitting numerical rule are available. Moreover, a prototype interpreter and a model for optimal rule sequence structure are designed. As a spin-off, software for highly accurate parameter conversion in LVS, based on NAGLIB, has been developed, and a tentative module was developed to transform the all pole specified LPC-spectra into pole-zero ARMA specified spectra by means of the Padé-approximation of polynomials. Unfortunately, a program to actually synthesize speech on the basis of the numerical rules has not been fully developed. Its incompleteness hampers the perceptual evaluation of the discovered rules.

The derivation of rules from a *particular* diphone set is certainly not the optimal approach. Instead of using *one* stylized version of *one* particular diphone set, the parallel use of several diphone sets ('diphone ensembles') is to be preferred from a statistical point of view. In the ALLODIF approach, much effort was done to rule the diphones themselves, instead of ruling all *allophones* in terms of the context. In the latter case, rule extraction could have taken place on the contexts /⋆ focus ⋆/, instead of the contexts /⋆ focus/ and /focus ⋆/. Such an approach had possibly led to an actual completely ruled-diphone synthesiser within the ALLODIF project. However, the correspondence between ALLODIF and ALSYS would be more difficult to establish. Moreover, some hard by-questions remain to be answered yet.

- One of such questions deals with the *convergence* of a rule set that is derived by analyses on diphone data towards the Nijmegen rule set. Under which circumstances are the rules in the Nijmegen set the concrete outcome of any numerical algorithm? Several reasons can be given why the development of such an algorithm will be a difficult task. Firstly, as we observed, the structure of the Nijmegen set does not precisely fulfill the specifications under which an (semi)automatic rule optimization is possible. Secondly, it is not clear whether the Nijmegen rule set is unique given its rule format. In other words: given the format of the Nijmegen synthsiser, a *lot of* rule sets produce identical acoustic output, and there is no simple criterion to decide which of the rule sets is to be prefered over the others.
- The parameter sets of the Eindhoven en Nijmegen synthesizers are different (appendix A). We face the technical problem, how to convert the spectral description derived from the Eindhoven diphone set to the description required for the Nijmegen synthesizer. The main difference is that the Nijmegen format allows us to define zeroes, and that both

sources are different.

The Nijmegen rule set can possibly be improved by matching individual Nijmegen rules to the diphone data. This idea has not been explored as it appeared to be a good alternative only in a late stage of the project. Moreover, such fitting is meaningless without restructuring the rule set at the same time: the fitted rule must not be affected by later rules, as the data-optimization does not account for such later fine tuning. In other words: that approach only works if the fitted rules have no daughters.

The problem of dealing with one diphone set only can be tackled by using larger labeled speech databases (e.g. the SPIN-database in development, or the databases available from SPEX). We observed that rule extraction can be done more reliable if more than one diphone version is available. If each allophone is stored in five versions, in all kinds of different allophone contexts, positions in the syllable and with different stress, a database should contain at least 30 minutes of speech material per speaker. For comparison, diphone sets contain only about 2 minutes of speech material. In this estimation, the context variation is systematic for contexts of length 3 (preceding phoneme, focus phoneme, following phoneme). Matrix entries corresponding to larger contexts will be filled only sparsely.

In the spirit of the preceding remarks, I would like to make the following points:

- Rule-extraction from data is possible, provided
  - the speech database is sufficiently rich (i.e. contains several versions of allophones in comparable contexts)
  - the rules have a very strict format
- (Semi-)automatic extraction of a rule *sequence* is possible if the sequence is structured along trees with conditions on the mother-daughter and sister-sister relation
- The algorithm to find rules is CPU-time consuming but allows us to extract '*knowledge*' from the output, contrary to any network approach
- A fast rule *interpreter* is required to optimize the rules perceptually and interactively.

These are points that may be of interest in case of continuation of this work. In further research, the best option seems to be a careful comparison between the IPO-KUN method and the COV-method, and to test them on a labeled speech database. As both methods seem to be promising, it is worth looking for a fruitful combination of ideas. My experience is that the implementation of a new idea in this area involves a lot of programming effort. In my opinion, the rule extraction and optimization activity can best be performed by a small group of researchers, supplied with a good software programmer. Apart from the perceptual improvement of rules, an algorithmic rule extraction is certainly within reach as soon as good labeled speech databases and fast processors are available.

Acknowledgement

This research was sponsored by the Dutch Program for the Advancement of Computer Science on Analysis and Synthesis of Speech SPIN-ASSP. Without such support, this fundamental research would never have been possible. Furthermore, I am indebted to Loe Boves for his ability to point to less lucky arguments. I also want to thank Henk Loman with whom I had inspiring discussions about rules and from whom I got a good impression of the difficulty of the rule optimization work. I must apologize to him for not being able to present him one single working speech rule. Last but not least I thank colleagues of IPO, for helping me getting familiar during the first months of my IPO-stay.

References

Allen, J., Hunnicutt, M.S., Klatt, D., Armstrong, R.C., and Pisoni, D.B. (1987). From text to speech: The MI-talk system. Cambridge University Press, Cambridge.

Bezooijen, R. van (1990). Evaluation of speech synthesis for Dutch: comparison of synthesis systems, intelligibility tests, and scaling methods. SPIN-ASSP report nr. 22.

Ten Bosch, L.F.M. (1989). From diphones to allophone rules. Proceedings AFN, Nijmegen. Vol. 13, p. 41-49.

Ten Bosch, L.F.M. (1990). Rule extraction for allophone synthesis. Proceedings of the ESCA-Workshop on Speech Synthesis, Autrans, France, sept. 1990. p. 17-20.

Ten Bosch, L.F.M. (1991). On the application of rule sets on linear symbol strings. (in preparation).

Chomsky, N., and Halle, M. (1968). The Sound Pattern of English. Harper & Row, New York.

Elsendoorn, B.A.G. (1982). Exploring the possibilities of speech synthesis with Dutch diphones. IPO Annual Progress Report 17, p. 63 - 65.

Golub, G.H., and Van Loan, C.F. (1983). Matrix computations. North Oxford Academic.

Loman, H., Kerkhoff, J., and Boves, L. (1989). A working environment for speech synthesis by rule. Proceedings AFN, Nijmegen. Vol. 13, p. 51-63.

Olaszy, G., Gordos, G., and Nemeth, G. (1990). Phonetic aspects of the Multivox text-to-speech system. Proceedings of the ESCA-Workshop on Speech Synthesis, Autrans, France, sept. 1990. p. 277-280.

Olive, J. (1990). A new algorithm for a concatenative speech synthesis system using an augmented acoustic inventory of speech sounds. Proceedings of the ESCA-Workshop on Speech Synthesis, Autrans, France, sept. 1990. p. 25 - 29.

Van Santen, J. and Olive, O. (1990). The analysis of contextual effects on segmental duration. Computer, Speech and Language. p. 359-390.

Wothke, K. (1990). From orthography to phonetic transcription in the German text-to-speech system Tetos. Proceedings of the ESCA-Workshop on Speech Synthesis, Autrans, France, sept. 1990. p. 219-223.

A The Eindhoven en Nijmegen parameter set

The Nijmegen allophone synthesizer is controlled by more speech parameters (37) than the Eindhoven synthesizer (14). For the spectral description of frames, the Nijmegen synthesizer can make use of zeroes (frequencies, bandwidths) as well. Moreover, 13 parameters deal with the temporal organisation of the targets and the intertarget movements. Also, the source in the Nijmegen synthesizer is much richer and more flexible than in Eindhoven. Below, we give a list of the Nijmegen parameter set.

1	form1	13	zero1	25	duur
2	bandbr1	14	bandz1	26	min duur
3	form2	15	zero2	27	max duur
4	bandbr2	16	bandz2	28	trans1
5	form3	17	zero3	29	trans2
6	bandbr3	18	bandz3	30	fltrs1
7	form4	19	zero4	31	fltrs2
8	bandbr4	20	bandz4	32	avtrans1
9	formfnp	21	formfnz	33	avtrans2
10	toonh	22	avampl	34	avstrans1
11	declin	23	avsampl	35	avstrans2
12	fnul	24	ahampl	36	ahtrans1
				37	ahtrans2

The Eindhoven parameter set is a subset of this Nijmegen set. It contains the 10 formant frequencies and bandwidths (Nijmegen parameters 1 to 8, plus fifth formant), the F_0 (parameter 10), the amplitude of the voiced source (parameter 22) and the unvoiced source (parameter 24), and the frame duration (parameter 25).

Table 5: The feature value specifications of the symbols used in the Nijmegen allophone synthesizer.

	pPtTkKcbdGC	fvszxgh	SZ"	mn~!*N	lr	jw	eiy@uoUOAaEI&	,.:
cons	+++++++++++	+++++++	+++	++++++	++	++	-------------	000
voc	-----------	-------	--0	------	--	--	+++++++++++++	--0
son	-----------	-------	--0	++++++	++	++	+++++++++++++	000
hoog	----+++--++	----++-	++0	--+--+	--	++	+++++--------	000
rol	-----------	-------	---	------	-+	--	-------------	00-
achter	----+++--+0	----+++	000	--0--+	--	--	----++-+++---	000
rond	++-----+--0	++-----	000	+-0---	+-	-+	--++++++----+	000
lang	-----------	-------	---	--0---	--	--	+--+-+---+---	00+
cont	-----------	+++++++	++0	------	++	++	+++++++++++++	00+
ant	++++---++--	++++--+	--0	++0-+-	-+	-+	------------0	000
mid	--++----+-+	--++--+	++0	-+0---	++	--	+--+-+++---++	000
nas	-----------	-------	---	++++++	--	--	-------------	00-
fric	-----------	+++++++	++0	------	--	--	-------------	00-
stem	-------+++-	-+-+-+0	-+0	++++++	++	++	+++++++++++++	000
seg	+++++++++++	+++++++	+++	++++++	++	++	+++++++++++++	--+
klem	-----------	-------	---	------	--	--	++++++++++++-	00-
asp	-+-+-+-----	------+	---	------	--	--	-------------	00-
	pPtTkKcbdGC	fvszxgh	SZ"	mn~!*N	lr	jw	eiy@uoUOAaEI&	,.:

B The Nijmegen feature specification

In table 5, the set of symbols ('segments') is shown which can be used to define the input symbol string for the Nijmegen synthesizer. Below each segment, its value for each of the 17 feature functions is specified. The plus-sign means that the property is present, a minus-sign points to its absence. A zero means that the function value is not specified.

For example, the vowels (ranging here from /e/ to /&/) are specified by [-cons] or by [voc]. Two column pairs are identical: /k/ has the same column as /c/, and ',' has the same column as '.'. Apart from these pairs, every symbol is characterized by at most five feature specifications (this is not a triviality). The phoneme that requires the least number of feature values in order to be specified is /r/: it is determined by [rol]. On the other hand, [rol] is a constant, except in the case of the symbol /r/. In other words, the symbol /r/ and the feature [rol] provide exactly the same information.
The mean number of minimally required specifications is about three. The minimal number of features required to specify a symbol is shown in table 6.

Table 6: Symbols (in the left column) and the minimal number of features required to uniquely specify them (right column).

symbols	nr.
r	1
PKh"mn~Nw&:	2
TCfvxgSZ!*le@uoO	3
pbdGszjiyUAEI	4
t	5

For fun, we present all the 14 possible five-feature-specifications of the symbol /t/ below. With less than five features, the /t/ cannot be uniquely specified. Observe that the features [stem] and [asp] are indispensible for the characterization of the /t/.

{hoog, rond, cont, stem, asp}, {hoog, rond, fric, stem, asp}, {hoog, cont, mid, stem, asp}, {hoog, mid, fric, stem, asp}, {achter, rond, cont, stem, asp}, {achter, rond, fric, stem, asp}, {achter, cont, mid, stem, asp}, {achter, mid, fric, stem, asp}, {rond, cont, ant, stem, asp}, {rond, cont, mid, stem, asp}, {rond, ant, fric, stem, asp}, {rond, mid, fric, stem, asp}, {cont, ant, mid, stem, asp}, and {ant, mid, fric, stem, asp}.

C The covariance method

The rule extraction method, developed by Van Santen and Olive from AT&T (Van Santen & Olive, 1990), refered to by 'covariance method' COV, differs from the IPO-KUN-method in two aspects.

1. COV deals with duration only
2. In COV, the translation from symbol classes such as [plos], [vowel] to numerical values is not fixed beforehand.

The first item means that they can use specific assumptions on the behaviour of duration as a function of context features, such as the 'joint-independence' property of factors. Joint-independence means implies that the following situation cannot occur:

	back	central	front
high	60	30	80
mid	30	20	60
low	40	35	50

as the *data ordering* in the first two colums does not correspond with the ordering in the third column.

Van Santen and Olive use four factors ('segment duration', 'phrasal location', 'number of syllables', and 'nuclear stress') and show that the phonetic parameter 'segment duration' has the joint independence property with respect to 'phrasal location', 'number of syllables', 'nuclear stress' and 'preceding segment'. From this, the authors conclude that segment duration can be modelled by a additive-multiplicative (AM) duration model, i.e. a model in which some of the four mentioned factors appear in one multiplicative factor and some factors occur additionally. A rule in the AM model has the following *rule skeleton*:

$$p = \prod_i m_i(f_i) + \sum_j m_j(f_j) \tag{10}$$

where p denotes a parameter, f_i and f_j denote factors (one of the four mentioned above; i and j running through two subsets of $\{1,2,3,4\}$), and the m's are *monotonous* functions that, for example, assign

no-stress	$\rightarrow$	20
secondary stress	$\rightarrow$	40
primary stress	$\rightarrow$	70

We observe that the rule skeleton has a multiplicative and an additive part. Here 'no-stress', 'secondary stress' and 'primary stress' are possible values of the factor stress with is one of the f_i and/or f_j. The functions m map these values to the numerical domain. These assignments are outcome of their covariance method as well.

The Klatt-model for the segment duration is *an example* of an AM model; there exist, however, no experimental evidence which points to the correctness of especially *this* exemplar of AM models. A basic property of the Klatt-model is that the factor 'segmental identity' is included in the addition part of the rule skeleton only. Van Santen and Olive show that another exemplar of these models provides a better fit to the data that they found in their duration database: They suggest to put the context factor in the additive part, and to place the factor 'segmental identity' in the multiplicative part.

With respect to the second item, viz. the non-determined translation from symbol classes to numerical values, we observe that the IPO-KUN-method applies fixed feature values as numerical translation of phonological classes. For example, the IPO-KUN-method in the case of vowels yields a table:

vowel category	height	'lowness'
high	2	0
mid	1	1
low	0	2

whereas the covariance-method starts with substituting

vowel category	height	'lowness'
high	a	d
mid	b	e
low	c	f

with the single condition that $c < b < a$ and $d < e < f$. This freedom allows the covariance method to find the same rules but in a more elegant format. A disadvantage is that the chosen assignment to a, b, c, d, e, f must in fact be fixed over other rules. Otherwise each rule should be preceded by a substantial redefinition of the default table. For this reason, I am a bit pessimistic about the advantage of such free assignments. Probably free assignments provide a neat way to formulate rules if one has to rule one parameter only and very reduced factor sets, as in Van Santen's case. For further details of his method I must refer to Van Santen & Olive (1990), as they are rather technical.

During the workshop on Speech Synthesis in Autrans (in September 1990), Van Santen and me had several long discussions on our methods. In fact, the IPO-KUN-method is a submethod of the COV-method, but the IPO-KUN-output is probably more interpretable for our specific purpose: the generation of rules for a rule set. Another point of particular interest is that the covariance method is able to find the rule skeleton in a systematic way. In the IPO-KUN-formalism, this skeleton is searched in a systematic way as well but the search strategy does not depend on the data, contrary to the situation in the COV-method. On the other hand, the search for that skeleton in the COV-method is rather sensitive to noise in the input data, as one crucial test is based upon the boolean $X = 0$ where X is some statistical parameter. The IPO-KUN-method can deal more easily with complex interactions.

Altogether, at this moment, it is difficult to decide what the best approach will be, since the IPO-KUN-method has not grown beyond the laboratory stage (the present state in December 1990).

We observe that the rule skeleton has a multiplicative and an additive part. Here 'no-stress', 'secondary stress' and 'primary stress' are possible values of the factor stress with is one of the f_i and/or f_j. The functions m map these values to the numerical domain. These assignments are outcome of their covariance method as well.

The Klatt-model for the segment duration is *an example* of an AM model; there exist, however, no experimental evidence which points to the correctness of especially *this* exemplar of AM models. A basic property of the Klatt-model is that the factor 'segmental identity' is included in the addition part of the rule skeleton only. Van Santen and Olive show that another exemplar of these models provides a better fit to the data that they found in their duration database: They suggest to put the context factor in the additive part, and to place the factor 'segmental identity' in the multiplicative part.

With respect to the second item, viz. the non-determined translation from symbol classes to numerical values, we observe that the IPO-KUN-method applies fixed feature values as numerical translation of phonological classes. For example, the IPO-KUN-method in the case of vowels yields a table:

vowel category	height	'lowness'
high	2	0
mid	1	1
low	0	2

whereas the covariance-method starts with substituting

vowel category	height	'lowness'
high	a	d
mid	b	e
low	c	f

with the single condition that $c < b < a$ and $d < e < f$. This freedom allows the covariance method to find the same rules but in a more elegant format. A disadvantage is that the chosen assignment to a, b, c, d, e, f must in fact be fixed over other rules. Otherwise each rule should be preceded by a substantial redefinition of the default table. For this reason, I am a bit pessimistic about the advantage of such free assignments. Probably free assignments provide a neat way to formulate rules if one has to rule one parameter only and very reduced factor sets, as in Van Santen's case. For further details of his method I must refer to Van Santen & Olive (1990), as they are rather technical.

During the workshop on Speech Synthesis in Autrans (in September 1990), Van Santen and me had several long discussions on our methods. In fact, the IPO-KUN-method is a submethod of the COV-method, but the IPO-KUN-output is probably more interpretable for our specific purpose: the generation of rules for a rule set. Another point of particular interest is that the covariance method is able to find the rule skeleton in a systematic way. In the IPO-KUN-formalism, this skeleton is searched in a systematic way as well but the search strategy does not depend on the data, contrary to the situation in the COV-method. On the other hand, the search for that skeleton in the COV-method is rather sensitive to noise in the input data, as one crucial test is based upon the boolean $X = 0$ where X is some statistical parameter. The IPO-KUN-method can deal more easily with complex interactions.

Altogether, at this moment, it is difficult to decide what the best approach will be, since the IPO-KUN-method has not grown beyond the laboratory stage (the present state in December 1990).

D A rule interpreter

In ALLODIF, a preliminary version of a rule-interpreter has been designed. The proposed rule set consists of (1) a part with basic definitions and (2) the rule sequence itself. The basic definitions involve *set definitions* such as (1) the universal symbol set, (2) the set definitions such as [plos], [vowel] etc., and (3) *parameter definitions* such as the meaning of the string 'F1', etc.

Set definitions have the e.g. following format:

```
[setname] = {e1/e2/e3/.../} or
[setname] = [plos]          or
[setname] = [plos.labial]   or
[setname] = {p/t/k}+[nas]   or
[setname] = [nas]-{m}
```

The first line gives an enumerative set definition by means of '{' and '}'. The second line an example of a declarative set definition (between '[' and ']'). For example, the set [plos] contains every segment for with the feature 'plos' has the value '1' or '+', and nothing more. These are the basic set construction methods.
All other sets can be constructed in either these two ways or by algebraic manipulations. Examples of these are presented in the last three lines. Three basic operations on sets are supported, represented by a dot, a plus sign and a minus sign, representing the intersection, the union, and the difference of sets, respectively.

A rule consists of

- a focus specification (i.e. *one* subset of the universal symbol set)
- a context specification (one list of sets before and one after the focus)
- action specifications

The focus and context specification is based upon the notion of sets. We discussed this possibility already in the section on the application in rule sets.

Action specifications involve the numerical treatment of anchor points that belong to all the time tracks of the phonetic parameters. These actions are specified in the postfix notation which is difficult to read for unexperienced humans but very easy to parse for a stack machine. In the postfix notation, $F_1 := F_1 + 200 + 0.1 \cdot F_2$ is written as

```
F1 200 + F2 0.1 * + -> F1
```

The great advantage of this notation is that interactively arbitrary actions can be defined and abbreviated. Moreover, the whole rule sequence is easy to parse and has a strict structure that allows the sequence to be a subject of an optimization algorithm. The use of *free variables* in rules is very easy to handle, for example in

```
F1 X + F2 0.1 * + -> F1
```

which means $F_1 := F_1 + x + 0.1 \cdot F_2$

where x can be modified interactively.