SVOX: The Implementation of a Text-to-Speech System for German

(2nd edition with corrected Appendix C, February 2002)

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Technical Sciences

presented by
CHRISTOF TRABER
Dipl. Informatik-Ing. ETH
born April 13, 1961
citizen of Thundorf (TG), Switzerland

accepted on the recommendation of
Prof. Dr. A. Kündig, examiner
Prof. emer. Dr. W. Guggenbühl, co-examiner

1995
Meinen Eltern und meinem Bruder gewidmet
Acknowledgments

First and foremost, I would like to thank Prof. Dr. A. Kündig and Prof. Dr. W. Guggenbühl for readily accepting to be examiner and co-examiner of the present dissertation. I would further like to thank them for their support of the speech synthesis project over many years.

I would like to thank the Swiss Telecom PTT for their generous support of the TTS project and for their patience in supporting such a large, interdisciplinary project. I also thank the Bundesamt für Bildung und Wissenschaft for sponsoring the synthesis project in the framework of COST 233.

I would like to thank all my former and present colleagues in the speech group, namely: Judith Krummenacher, Ruth Rothenberger, Hans Huonker, Thomas Russi, Karl Huber, Marcel Riedi, Peter Sempert, Carlo Bernasconi, Hans Forster, Beat Pfister, Hans-Peter Hutter, Schamai Safra, Daniel Christnach, Christoph Messner, and Gunnar Lehtinen, for the easy, yet industrious atmosphere which they created, and for many humorous, inspiring, and fruitful discussions and coffee breaks.

I would like to thank Beat Pfister for leading the speech processing group and for his innumerable administrative works. I especially thank him for his enormous efforts to keep the speech group alive in the hard times after Prof. Guggenbühls retirement. The speech group would otherwise probably have ceased to exist.

I am especially obliged to Karl Huber and Thomas Russi, who were my linguistics teachers and who will always be examples to me. They taught me how to think about speech and language processing in a
scientific manner. Without their presence and without their influence on me, the SVOX system would probably have ended up in quite a different shape.

I wish to thank Karl Huber, Marcel Riedi, and Beat Pfister for carefully reading the draft of this thesis and for helpfully commenting on it, and I am very grateful to Michael Hunt and Barbara Waldvogel for correcting my English.

I am indebted to numerous students who have directly or indirectly contributed to the design and implementation of the SVOX system.

Ich danke meiner Familie, allen voran meinen Eltern Theres und Walter und meinem Bruder Markus, die mich immer unterstützt haben und die mir ermöglichten, das Studium und die Dissertation in freier Wahl durchzuführen.

I wish to thank my colleague Andreas Schilling for many years of close friendship, for critical and helpful remarks and encouragement, for his physical training lessons (Kondi), and for being patient when I was not in the best of moods when writing the dissertation.

I would like to thank Gabrielle Gross, Daniel Mantovani, Thomas Lenggenhager, Roland Köppel, and Marcel Neuhausler for many hours and days of distraction and traveling adventures.

Finally, I would like to thank the Zurich forests and the Zurich trams for their help in writing this report, and I thank the Entropic company for providing the speech group with a set of inspiring, high-quality yo-yos.

## Contents

### Kurzfassung

### Abstract

### List of Abbreviations

### 1 Introduction

1.1 Purpose and Scope of this Thesis ........................ 19

1.2 Applications of TTS Systems ........................... 20

1.3 Recent History of the ETH TTS Project .................. 22

1.4 The Interdisciplinarity of TTS Research ................ 24

1.5 Linguistic Terminology ................................. 25

1.5.1 Linguistic Levels and Subdisciplines ................ 25

1.5.2 Linguistic Units .................................. 27

1.6 Structure of this Thesis ................................. 30

### 2 TTS Synthesis and the SVOX Architecture

2.1 Introduction ........................................... 33
2.2 Major Problems of TTS Synthesis .......................... 34
  2.2.1 Generation of Speech Sounds .......................... 34
  2.2.2 Derivation of Phonetic Segments ....................... 36
  2.2.3 Derivation of Prosody ................................. 41
2.3 The Development of a TTS Model ............................ 48
2.4 The SVOX Architecture .................................... 50
  2.4.1 Design Considerations .................................. 50
  2.4.2 Structure of the SVOX System .......................... 52
  2.4.3 Types of Processing .................................... 55
  2.4.4 Implementation Overview ............................... 56
2.5 Other TTS Architectures .................................... 57

3 Syntactic and Morphological Analysis ......................... 59
  3.1 Syntactic and Morphological Analysis in a TTS System .... 59
  3.2 Morpho-Syntactic Analysis in the SVOX System ............ 61
    3.2.1 Architecture Overview ................................ 61
    3.2.2 Original Realization .................................. 64
    3.2.3 Modifications in the Morpho-Syntactic Analysis ..... 64
    3.2.4 Definite Clause Grammars (DCGs) ..................... 66
  3.3 Formalism of the Morpho-Syntactic Analysis ............... 66
    3.3.1 Feature Terms ....................................... 67
    3.3.2 Unification ......................................... 68
    3.3.3 Grammars ........................................... 69
    3.3.4 Lexicons ........................................... 71

4 Grammar-Based Grapheme-to-Phoneme Mapping ................. 103
  4.1 Introduction ............................................. 103
  4.2 Abbreviations ........................................... 104
  4.3 Numbers ............................................... 107
  4.4 Unknown Words ......................................... 111
    4.4.1 Introduction ......................................... 111
    4.4.2 Pronunciation of Novel Words in SVOX ............... 112
  4.5 Summary ............................................... 119

5 Accentuation and Prosodic Phrasing ........................... 121
  5.1 Introduction ............................................. 121
  5.2 Syllabification ......................................... 122
  5.3 Accentuation ........................................... 123
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1</td>
<td>Introduction</td>
<td>123</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Accentuation Principles</td>
<td>124</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Accentuation Algorithm in SVOX</td>
<td>125</td>
</tr>
<tr>
<td>5.4</td>
<td>Prosodic Phrasing</td>
<td>132</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Introduction</td>
<td>132</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Phrasing Principles</td>
<td>135</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Phrasing Algorithm in SVOX</td>
<td>136</td>
</tr>
<tr>
<td>5.5</td>
<td>Accent Normalization</td>
<td>140</td>
</tr>
<tr>
<td>5.6</td>
<td>Phonological Representation</td>
<td>144</td>
</tr>
<tr>
<td>6</td>
<td>The Phono-Acoustical Model</td>
<td>147</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>147</td>
</tr>
<tr>
<td>6.2</td>
<td>Segmental Phonology-to-Phonetics Mapping</td>
<td>148</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Reduction of Long Vowels</td>
<td>148</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Phone Substitutions</td>
<td>149</td>
</tr>
<tr>
<td>6.3</td>
<td>Diphonization and Duration Control</td>
<td>150</td>
</tr>
<tr>
<td>6.4</td>
<td>Fundamental Frequency Control</td>
<td>153</td>
</tr>
<tr>
<td>6.5</td>
<td>Speech Signal Generation</td>
<td>154</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Diphone Synthesis</td>
<td>154</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Diphone Synthesis in the SVOX System</td>
<td>154</td>
</tr>
<tr>
<td>6.6</td>
<td>The TD-PSOLA Method for Prosody Modifications</td>
<td>156</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Introduction</td>
<td>156</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Preparatory Speech Signal Labeling</td>
<td>156</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Signal Analysis</td>
<td>157</td>
</tr>
<tr>
<td>6.6.4</td>
<td>Signal Resynthesis with Modified Prosody</td>
<td>159</td>
</tr>
<tr>
<td>6.6.5</td>
<td>Implementation Details</td>
<td>160</td>
</tr>
<tr>
<td>7</td>
<td>Neural-Network-Based Fundamental Frequency Control</td>
<td>165</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>165</td>
</tr>
<tr>
<td>7.2</td>
<td>$F_0$ Generation in SVOX</td>
<td>167</td>
</tr>
<tr>
<td>7.3</td>
<td>A Neural Network Architecture for $F_0$ Prediction</td>
<td>168</td>
</tr>
<tr>
<td>7.4</td>
<td>The Training Corpus</td>
<td>172</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Text Material</td>
<td>172</td>
</tr>
<tr>
<td>7.4.2</td>
<td>$F_0$ Treatment</td>
<td>173</td>
</tr>
<tr>
<td>7.4.3</td>
<td>Phonological Transcription</td>
<td>175</td>
</tr>
<tr>
<td>7.5</td>
<td>Input and Output Coding</td>
<td>176</td>
</tr>
<tr>
<td>7.5.1</td>
<td>Window Technique</td>
<td>176</td>
</tr>
<tr>
<td>7.5.2</td>
<td>Input Coding</td>
<td>178</td>
</tr>
<tr>
<td>7.5.3</td>
<td>Output Coding</td>
<td>181</td>
</tr>
<tr>
<td>7.6</td>
<td>NN Training</td>
<td>181</td>
</tr>
<tr>
<td>7.6.1</td>
<td>Training Algorithm</td>
<td>181</td>
</tr>
<tr>
<td>7.6.2</td>
<td>Training Experiments</td>
<td>184</td>
</tr>
<tr>
<td>7.7</td>
<td>Results</td>
<td>186</td>
</tr>
<tr>
<td>7.8</td>
<td>Formal Quality Evaluation</td>
<td>188</td>
</tr>
<tr>
<td>7.8.1</td>
<td>Experiment</td>
<td>188</td>
</tr>
<tr>
<td>7.8.2</td>
<td>Results and Discussion</td>
<td>189</td>
</tr>
<tr>
<td>7.9</td>
<td>Generalization Behavior</td>
<td>192</td>
</tr>
</tbody>
</table>
Kurzfassung


Aus dem in der morpho-syntaktischen Analyse abgeleiteten Syn-
taxbaum wird mittels regelbasiertem Verfahren, das auf modifizierten Verfahren aus der linguistischen Literatur beruhen, die phonologische Repräsentation einer Aussage erzeugt, die aus der phonetischen Umschrift der Wörter, der Akzentverteilung und der Phrasierung besteht.


Abstract

The present thesis describes the implementation of the text-to-speech system SVOX for the German language. The linguistic concept, the overall architecture, and several components of this system (diphone synthesis, morpho-syntactic analysis, and phone duration control) were already treated in dissertations of other authors and formed prerequisites for this work. The newly treated issues in the framework of the present dissertation are mainly grapheme-to-phoneme mapping in cooperation with the morpho-syntactic analysis, accentuation and prosodic phrasing, and fundamental frequency control by means of neural networks.

The first part of the thesis describes a re-implementation of the morphological and syntactic analysis, in which a slightly extended DCG (definite clause grammar) formalism is applied. It is shown how also abbreviations, digit sequences, and unknown, i.e., not lexically stored, words can be analyzed and converted into the pronunciation form, without any essential extensions of the concept or the formalism of the regular morpho-syntactic analysis. All these grapheme-to-phoneme mappings are done by means of grammars which describe the internal structure of graphemic objects. A phonetic transcription is assigned to the components of these objects according to lexicon entries, and the concatenation of the individual transcriptions yields the transcription of the entire object. The conversion of the graphemic representation into the phonetic transcription is also reversible due to the uniform declarative implementation.

From the syntax tree derived in the morpho-syntactic analysis, the phonological representation of an utterance is generated by applying
rule-based methods, which are modified versions of methods given in the linguistic literature. The phonological representation consists of the phonetic transcription of the words in the utterance, the distribution of accents, and the prosodic phrasing.

The synthetic speech signal is generated by concatenation of phone transition elements (diphones). Phone durations and the fundamental frequency contour are derived from the phonological representation by means of statistical methods. For fundamental frequency control a recurrent neural network is applied. Natural speech signals are used to train such neural networks to map the phonological representation of utterances onto the corresponding fundamental frequency contour. In the training process internal representations of the rules of this mapping are generated which are also able to predict fundamental frequency contours of high quality for new phonological representations.

List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>American standard code for information interchange</td>
</tr>
<tr>
<td>ATN</td>
<td>Augmented transition network</td>
</tr>
<tr>
<td>CTS</td>
<td>Concept-to-speech</td>
</tr>
<tr>
<td>DCG</td>
<td>Definite clause grammar</td>
</tr>
<tr>
<td>$F_0$</td>
<td>Fundamental frequency</td>
</tr>
<tr>
<td>FST</td>
<td>Finite-state transducer</td>
</tr>
<tr>
<td>IPA</td>
<td>International Phonetic Association</td>
</tr>
<tr>
<td>LPC</td>
<td>Linear predictive coding</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer perceptron</td>
</tr>
<tr>
<td>NN</td>
<td>Neural network</td>
</tr>
<tr>
<td>TD-PSOLA</td>
<td>Time-domain pitch-synchronous overlap-add</td>
</tr>
<tr>
<td>TTS</td>
<td>Text-to-speech</td>
</tr>
<tr>
<td>UTN</td>
<td>Unification-based transition network</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Purpose and Scope of this Thesis

The work presented in this thesis is the implementation of a so-called *text-to-speech synthesis system* for the German language, i.e., a system that converts written orthographic German text (in computer-readable form) into corresponding artificial speech signals, which are represented as digitized audio signals. (The term “text-to-speech” is usually abbreviated to “TTS”, and this abbreviation will be used frequently hereafter.)

The purposes of this thesis are several: first and foremost, it is the report on the work that the author has carried out to take his doctor’s degree; second, the TTS system presented here is described in enough detail to enable other researchers to verify the (in some cases new) methods and procedures applied therein; third, it should help new researchers in our group to carry on this work; and fourth, although this thesis cannot be a tutorial on TTS synthesis, it is a case study of the development of a specific TTS system and it could therefore help newcomers in the field to get started with their own work.

It is necessary here to very clearly state what the author’s own contributions were to the realization of the TTS system described in this report (which was called “SVOX” from 1992 onward). Several people have contributed to the design and realization of the system and the
1.2 Applications of TTS Systems

Text-to-speech systems aim at imitating the human ability to read aloud written texts, where “text” usually stands for computer-readable sequences of characters (e.g., ASCII coded text). In some applications, TTS systems are combined with an OCR (optical character recognition) device, which translates optical images into computer-readable text. Such an optical front-end creates a real “reading machine” (used, for instance, by blind people), which is probably the most often cited application of TTS synthesis. In other cases an interface like the Blis symbol notation system, which is a shorthand notation for words in a text, or automatic word prediction capabilities allow an easy and fast entering of text into a TTS system [CG86].

The afore-mentioned machines serve as aids for vocally and/or manually impaired persons. Apart from its use in aids for the handicapped, the application range of pure TTS synthesis is rather limited. Among such applications are one-way information communication systems over the telephone line (e.g., weather forecasts, news, or market reports), the reading of personal electronic mailbox contents over the telephone line, and proofreading aids\(^1\). Apart from aids for the handicapped, however, hardly any professional (and profitable) TTS applications seem to be currently in use.

It is widely acknowledged that the greatest potential for future speech synthesis applications will lie in dialog systems, i.e., in systems that are capable of a two-way speech communication between man and machine. Dialog systems in general will incorporate speech recognition (or a mixture of speech recognition and input devices such as dual tone multifrequency signaling, DTMF), speech synthesis, a speech dialog management component (to keep track of the dialog state and to generate machine answers), and a problem-solver component (for example, database queries in train schedules). Such prototype dialog systems of remarkably high quality already exist. However, the synthesis component of a dialog system should ideally not be a TTS synthesis: Since the dialog system “knows” all the details of the output utterance (including semantic details of each word, the focus of the utterance, style of pronunciation etc.), it would be useless to first convert an abstract internal representation (or so-called “concept”) into an orthographic text form, because the subsequent TTS system would have to “guess” the original intentions and meaning of the utterance again.\(^2\) Therefore, in connec-\(^1\) It is in fact quite surprising to hear how well spelling errors, which could easily be overlooked in ordinary proofreading, are noticeable when uttered by a good TTS system, even if one listens to the synthetic speech with only moderate attention. Because of this, synthetic speech is also highly impaired in other TTS applications when typing errors are encountered.

\(^2\) In current dialog systems, instead of using genuine speech synthesis, speech is often generated by the concatenation of a limited inventory of longer parts of natural (human) utterances. This is mostly due to the still limited quality of today’s available speech synthesis systems, and it is to be expected that future dialog systems will make full use of speech synthesis, because of the numerous advantages it
tion with dialog systems, the appropriate form of speech synthesis is a so-called concept-to-speech (CTS) system, which directly generates utterances from abstract semantic/pragmatic concepts without necessarily generating an orthographic representation at all.\textsuperscript{3}

Nevertheless, TTS systems are not superfluous since large parts of TTS and CTS systems can be shared, as will become clear in the following sections, and most of the knowledge acquired in TTS research will be exploitable also in CTS systems.

1.3 Recent History of the ETH TTS Project

A detailed account of the ETH TTS project until 1991 has been presented in the preface of [Hub91]. A summary of this and of the most recent new developments of the project is given in this section.

The first phase of the TTS project was dedicated to research on speech signal generation methods and the preparation of an LPC diphone inventory for German by Hubert Kaeslin, which was finished in 1985 [Kae85].\textsuperscript{4} Then, from 1985 to 1991, an interdisciplinary research group of engineers and linguists worked on syntactic and morphological analysis, prosody control, and the realization of a TTS prototype. The members of that group were: Karl Huber, Thomas Russi, Hans Huonker, Judith Krummenacher, Ruth Rothenberger, Peter Sempert, and the author. Beat Pfister was the head of the group and coordinated the different parts of the research project.

In 1991, a demonstration prototype was finished which was able to perform the full text-to-speech synthesis for weather forecasts. The forecast texts were automatically transferred to the system from the Schweizerische Meteorologische Anstalt (SMA; Swiss meteorological institute), and the synthesis of the latest forecast was carried out five times a day. This prototype application was presented at the 1991 science exhibition “Heureka” in Zurich. At that time, the (nameless) system consisted of two completely separate programs, one for the syntactic and morphological text analysis (written in Lisp; cf. [Rus90]) and one for prosody control and signal generation (written in Prolog and Modula-2). The system had a tremendous size (it was possibly the world’s largest and slowest TTS system), but being mainly a research instrument rather than a cheap commercial application, this did not cause real difficulties. The most intricate problem at that time was that for unknown words (i.e., words not stored in any of the system’s lexicons) no pronunciation could be found, which caused the synthesis of the whole sentence containing that word to fail.\textsuperscript{5}

From 1991 to 1994, the research team of the synthesis project was reduced to two people (Marcel Riedi and the author). The most obvious change to the system during that period was the addition of a procedure to obtain the pronunciation of novel words (cf. Chapter 4). In order to realize this procedure in the intended way, it became necessary to re-implement the entire syntactic and morphological analysis (cf. Chapter 3). This re-implementation also led to a smaller and better manageable system without the loss of much flexibility. In the course of the re-implementation, Sascha Braver, a computer linguistics student, implemented a top-down chart parser and also redesigned certain parts of the sentence grammar [Bra94]. Moreover, a lexicon editor was added [SN93], which simplified the task of acquiring new lexicon entries (cf. Chapter 3). With the help of this editor, the morphemes of the basic German vocabulary (approximately 5000 entries) were added by the author to the already existing morpheme lexicon.

Another major change in this period was the application of the TDP-SOLA (time-domain pitch-synchronous overlap-add) method in combination with diphone synthesis (cf. Chapter 6), which considerably increased the quality of the artificial speech signal.

\textsuperscript{3}For the surprisingly limited domain of weather forecasts, it was no problem to collect almost the entire vocabulary. However, problems still arose due to frequent typing errors; even slightly misspelled words were regarded as “unknown”, and no pronunciation could be found.
The current TTS system, now called SVOX, is still quite large and slow, but on computers as powerful as modern workstations it is able to synthesize texts with few novel words in real time. The system is currently being run by the Swiss PTT as a prototype application for reading weather forecasts over the telephone line. It is also used as a preliminary synthesis component of a dialog system which is under development in the speech processing research group.

1.4 The Interdisciplinarity of TTS Research

A great deal of the effort of building a TTS system is devoted to adopting ideas, techniques, and algorithms from rather different scientific disciplines and to linking together pieces of different nature into an overall system of plausible and uniform structure. From this point of view, to build a TTS system is, to a large extent, genuine engineering work.

However, difficulties still arise today when disciplines as different as pure linguistics and signal processing must be combined. Great differences exist in the ways of thinking, the terminologies, and the methodologies, even between "pure linguistics", "computer linguistics", and phonetics, and very often desirable dialogues or even collaboration do not take place. Probably the worst situation is then that of the engineer in his task to build a language and speech processing system which should integrate parts from all these different worlds.

Within these conflicts encountered in realizing a speech processing system, one must find feasible solutions for the linguistic problems at hand. This means, however, that it will not always be possible to make use of the latest linguistic theories, which are not yet complete enough or which rely on information that is beyond the reach of today's speech processing systems.¹

For the TTS project presented in this thesis, the general position adopted by the researchers was therefore to mostly rely on simple, well-established linguistic methods and formalisms. The whole system should, however, still be "as linguistic as currently feasible", and it should not just be a small system that could somehow convert text into speech sounds with as little effort as possible.

1.5 Linguistic Terminology

Since all language processing systems involve linguistics to a very high degree, it is necessary to briefly introduce the most important linguistic terms used in the remainder of this report and to define in what sense these terms will be used. In the first part of this section, linguistic levels and subdisciplines will be characterized, and in the second part, linguistic units will be defined. More specific linguistic terms will be introduced when needed in later chapters.

1.5.1 Linguistic Levels and Subdisciplines

Syntax: The description and analysis of well-formed sentence structures of a language: The sentences "I went to the cinema," and "The green dream stood over the liver of your chimney," are syntactically well-formed English sentences. Syntax assigns a hierarchical structure to a sentence, in which syntactic constituents of one level are grouped together to form constituents of higher levels. These constituents are parts of a sentence that can readily be replaced by similar parts built from other words. For example, in the above sentence "I went to the cinema," "I" can be replaced by "you", "the young man" etc., and "to the cinema" can be replaced by "into the house", "up the hill", but the word "went" can hardly be replaced by other verbs (e.g., "sang", "read") without also changing the remainder of the sentence. Therefore, a plausible syntactic structuring of the sentence is ((I) (went (to the cinema))).

Morphology: In analogy to syntax, morphology describes how words can be built from minimal meaningful elements of a language, the so-called morphemes. For example, the English word "rebuilding" is built from the prefix morpheme "re", the verb stem morpheme "build", and the inflection morpheme "ing".
Semantics: The investigation and treatment of the meaning of words and sentences. For example, of the two syntactically well-formed sentences “I went to the cinema.” and “The green dream stood over the liver of your chimney.” only the first has a well-defined meaning.

Pragmatics: The investigation and treatment of the intentions and the communicative function of words, sentences, and texts. For example, the sentence “I went to the cinema.” is uttered quite differently if it represents a friendly answer to a purely informative question (e.g., “What did you do yesterday evening?” – “I went to the cinema.”) or if it is shouted out in a man’s desperate attempt to convince a police officer of his innocence (e.g., “I don’t believe you. You’re really trying to tell me that you were not at home at 9 o’clock yesterday evening?” – “I went to the cinema.”).

Grammar: In the present thesis, “grammar” denotes a collection of rules which define the structure of linguistic objects. Grammars in this sense are very often formulated by means of a set of production rules, which were introduced by the theory of formal languages [Cho57]. In the linguistic framework of [Cho65], “grammar” is used in a wider sense and comprises a semantic, a syntactic, and a phonological component.

Lexicon: In the linguistic sense, “lexicon” is the collection of words and morphemes which belong to a particular language. In the present thesis, “lexicon” will denote a collection of linguistic items (e.g., morphemes, words, or consonant clusters) that are used as basic elements in a grammar.

Phonology: The investigation and treatment of minimal necessary distinctions that separate two spoken utterances with different meanings from one another. For example, the utterance “the wheel arrived” differs from “the veil arrived” only in the manner of articulation of the first sound of the second word (rounded bilabial sound vs. labio-dental sound); the utterances “the black bird” and “the blackbird” differ only in their stress patterns which are realized as different rhythmic and melodic patterns.

Phonology attempts to describe utterances by sequences of such minimal distinctions and to relate these descriptions to other linguistic representations (mainly syntactic, semantic, and phonetic representations).

Phonetics: The investigation of the actual production and perception of spoken utterances. In contrast to phonology, which is primarily interested in the characterization of the minimal necessary distinctions between different utterances, phonetics treats thevariability and individuality of actual realizations of utterances. Whereas phonology uses symbolic representations, phonetics is faced with numeric acoustic data and the application of statistical methods is therefore often required in phonetics.

Prosody: The investigation of the manner of articulation of a speech sound sequence in terms of intonation (or melody), rhythm, and loudness, which are phonetically manifested in the acoustic properties pitch (fundamental frequency, henceforth termed F0), speech sound duration, and speech signal intensity. There are phonetic as well as phonological aspects to prosody. For instance, in some languages certain kinds of questions are marked by a rising F0 pattern at the end of the utterance, whereas statements are characterized by an F0 drop. This binary distinction (fall or rise) is of phonological nature. The description of how large F0 rises and falls are on average and how they depend on different speakers is a question of phonetic nature.

Prosodic elements are often also termed suprasegmental features, since these elements (e.g., melodic patterns) extend over several speech segments (the speech sounds).

1.5.2 Linguistic Units

Grapheme, graph, allograph: The grapheme is the smallest unit of the written language which distinguishes different written words. It is an abstract representation of the class of all possible graphic signs denoting the same distinctive element. Graphemes or grapheme sequences are written between “<” and “>”.

An individual realization of a grapheme is a graph. Graphs belonging to the same grapheme are called allographs of this grapheme. For example, “a”, “”, “a”, and “a” are allographs of the same grapheme <a>.

Phoneme, phone, allophone: The phoneme is the smallest unit of the spoken language that distinguishes different meanings of utterances. For example, the English words “my” and “sigh” are
distinguished by a different first phoneme. The phoneme is an abstract unit which can be described as the minimal set of distinctive articulatory or acoustic features that are needed to distinguish different utterances that differ in only one phonetic segment. Traditionally, phonemes are represented as vectors of binary features, which are usually represented by phonemic symbols between ‘/’ and ‘\’. For example, the first phoneme of the word “my” is characterized by the features [-vocalic, + consonantal, - compact, + nasal, + grave], and the representation is “\m/”. Phoneme inventories differ from one language to another. For example, the degree of aspiration of unvoiced plosive consonants (’p’, ‘t’, and ‘k’) is non-distinctive in German, but it is distinctive in e.g., the Thai language.

A phone is a minimal segment of an utterance, which is one particular realization of a phoneme. Phones and phone sequences are denoted by phonetic symbols between ‘[’ and ‘]’. Phones that belong to the same phoneme are called allophones of this phoneme. For example, the German phones [a] and [e] are realizations of the different phonemes /a/ and /e/. The German phones [g] and [f] are allophones of the phoneme /r/.

Text: A sequence of sentences belonging to the same topic.

Sentence: A sequence of words belonging to the same syntactically and semantically closed statement or question. The sentence in a linguistic sense is less well-defined than in the orthographic sense. However, in computer linguistics, it is common to refer to the sentence in the orthographic sense, which is usually terminated by ‘,’ ‘!’, or ‘?’. This will be the definition adopted in this report.

Word: A morphologically and semantically closed unit that denotes a specific item (object, action, property) of the world. Words can be realized in a graphemic and in a phonemic and phonetic form. The term “word” is highly problematic. For example, the orthographic German word “Schulsgebäude” is “school building” in English, i.e., written in two orthographic words. In this thesis, “word” will always denote a linguistic unit which corresponds to the German orthographic word.

Morpheme, morph, allomorph: The morpheme is the smallest unit of language (in general smaller than a word) carrying a well-defined meaning. Morphs are actual (graphemic and phonemic) realizations of morphemes. Morphs that are realizations of the same morpheme are called allomorphs of this morpheme. For example, the German stem morpheme “baum” (“tree”) is realized by the two allomorphs <baum> /baum/ for singular forms and <bäm> /bäym/ for plural forms. The English word “reconsider” can be split up into the morphs <re>, <consider>, and <ed>, which are realizations of the prefix morpheme “re”, the stem morpheme “consider”, and the past tense ending morpheme.

Syllable: The syllable is a unit of the spoken language which is composed of a voiced center (the syllabic nucleus: a vowel, a diphthong, or a sonorant consonant) with an intensity (or sonority) maximum and optionally some consonants around it, such that each syllable is delimited by two intensity minima. It is much easier to define the syllable center than the syllable boundaries. An utterance can be split up into a sequence of adjacent syllables (except for speech pauses). The syllable is the most important unit for prosody.

Accent, stress: Accent or stress denotes prominence of an uttered syllable over other syllables. This prominence can be realized to different degrees. For example, in the English sentence “I saw the book lying on the table,” “book” and “table” carry more prominence than “on” or “saw” and are therefore more strongly accented. In the linguistic literature, the terms “accent” and “stress” are sometimes used with different meanings (e.g., “stress” more referring to durational lengthening of syllables and “accent” denoting prominence marked by major pitch changes). In the present thesis, the terms will be used completely interchangeably, although the term “stress” will be used more often in connection with individual words, and “accent” more often in connection with sentence accentuation.

Phrase, prosodic phrase: Phrases are parts of an utterance which in general consist of several words that are uttered as rhythmic and/or melodic units. For example, it is plausible to split up the English sentence “I saw the large tree behind the house” into the phrases “I saw / the large / tree / behind the / house”, but not into “I / saw the large / tree behind the / house”. The boundaries between phrases may be indicated by pauses, by lengthening of the final syllable(s) before the boundary or by certain melodic patterns.
The term *prosodic phrase* is sometimes used in the literature to distinguish the phrase in the prosodic sense from the phrase in the syntactic sense (as in “noun phrase,” “verb phrase”).

**Phonetic transcription, phonological representation:** The *phonetic transcription* is a standardized, canonical description of the pronunciation of words as found in phonetic dictionaries (in German for instance in [Dud74]). This transcription is usually somewhat more precise than a genuine phonemic transcription, but does not distinguish many allophonic variants. For example, [Dud74] distinguishes between two /t/ allophones, namely [t] and [ɾ], but not between the different degrees of aspiration in the /t/ phoneme at the beginning and end of the German word “Tat” (“act”).

Standardized pronunciation forms play an important part in the realization of TTS systems. In this thesis, the term *phonological representation* of an utterance will frequently occur and will denote an abstract, minimal representation of the segmental and prosodic features of an utterance. Due to this definition, the phonological representation should actually contain (besides accentuation and phrasing information) the *phonemic transcription* of the words in the utterance. For practical reasons, however, the “phonological” representation in the SVOX TTS system contains the standardized phonetic transcription instead of a purely phonemic representation.

### 1.6 Structure of this Thesis

The following chapters of this thesis will describe in detail the current implementation of the SVOX TTS system. Chapter 2 will introduce the problems associated with TTS synthesis and the general designs of TTS systems, and an overview of the SVOX architecture will be given. The subsequent chapters will present the different TTS processing stages, and in particular, Chapters 4, 5, and 7 will present the author’s major contributions to the TTS problem. Chapters 3 and 6 mainly serve to explain the environment into which the author’s work had to fit and to give an impression of the implementation of a TTS system as a whole. The different TTS stages (and thus, the different chapters) will not be presented in chronological order of development but more or less in the same order as for the stages of conversion of an input text into a corresponding speech signal.

The appendices contain fully detailed descriptions of certain aspects of the SVOX system.
Chapter 2

TTS Synthesis and the SVOX Architecture

2.1 Introduction

This chapter presents a short introduction to the general linguistic problems faced in all TTS systems, summarizes the considerations that led to the design of the current SVOX system, and presents an overview of its architecture, which is mainly due to Karl Huber [HHP+ 87, Hub91]. It should be mentioned here that the architecture of the current SVOX system is still nearly identical to the original 1986 design, although at that time it was not at all clear how the different modules should be realized. This design stability was one of the main prerequisites for the realization in parallel of different parts of the system by different researchers.

In the subsequent sections and chapters, the term “ETH TTS project” will be used to denote the TTS project of the ETH speech processing group from 1986 to 1991, and the term “SVOX system” will be used to denote the further development of the same TTS system, but from 1991 onward, starting with the re-implementation of the syntactic and morphological analysis.
2.2 Major Problems of TTS Synthesis

In order to establish a model of a TTS system, the general problems encountered in any such system must be known. A brief survey of these problems and information requirements is given in this section. An excellent overview of general TTS issues in connection with the development of TTS architectures for English can be found in [Kla87].

2.2.1 Generation of Speech Sounds

The primary prerequisite of any speech synthesis is a speech signal generator, i.e., an apparatus or a piece of software capable of producing sequences of sounds similar to human speech. Historically speaking, this is where speech synthesis development started. The approaches to the synthesis of speech segments are usually grouped into three classes:

- The *articulatory synthesis* tries to physically model the human process of articulation by means of a physical model of the human vocal tract and a system of rules that maps a sequence of allophones onto movements of the articulators. Although this is a natural and logical approach, it is extremely difficult to realize, and the results achieved so far are clearly of low quality compared to other approaches to segmental synthesis. Nevertheless, articulatory synthesis is extremely flexible and the most explicit and explanatory type of synthesis.

- The *signal models* do not care about how speech sounds are produced in the human vocal tract. Instead, they "only" try to model the relationship between allophones and the corresponding relevant properties of the speech signal. The most important of these properties are the mutual positions and temporal courses of the so-called formants, i.e., of the most prominent peaks of the speech signal spectrum. Signal models are therefore usually realized as *formant synthesizers*, in which the course of each formant (up to about the fifth formant) is controlled by a time-varying digital filter. All filters together are applied to a source signal which models turbulences of the air flow in the human vocal tract by a noise signal and the vibration of the vocal folds by a periodic pulse train.

The parameters of the formant filters are most commonly controlled by rules that translate allophone sequences into the filter parameter tracks (e.g., [AHK87]). Formant synthesizers represent a highly flexible approach to speech signal generation, but they require a large amount of research work to produce intelligible, natural-sounding speech. Nevertheless, there exist formant synthesizers which produce speech of very high naturalness (e.g., [AHK87, Co92]).

- The *concatenative synthesis* simply combines pieces of natural human speech into new utterances, i.e., the generated speech signal is not entirely synthetic. In order to be able to realize arbitrary utterances, the inventory of basic speech elements must be chosen appropriately. The currently most popular concatenative synthesis is the *diphone synthesis*, in which the basic elements are diphones, i.e., speech parts ranging from the middle of one speech sound to the middle of the next (e.g., [CM89, Kae86]). The concatenative approach currently produces the highest quality of synthetic speech, and a concatenative synthesis system can be built with a relatively small amount of work. However, the approach is rather unflexible. Each new voice or each new timbre requires establishing of a new speech element inventory.

One of the major problems of concatenative synthesis is the modification of duration and pitch of the speech elements without changing the spectrum of the signal. These modifications are necessary to generate the prosody of the synthetic speech. Two standard solutions of this problem are LPC synthesis [Mak75, Kae86] and the newer and better TD-PSOLA synthesis [CM89, MC90].

Diphone synthesis will be further discussed in Chapter 6.

In order to drive the speech generation component, the sequence of *sounds* to be produced must be known as well as the *prosody* of this sequence, i.e., the manner in which the sounds are to be produced (at what pitch, with what duration and with what intensity). The following two sections discuss these two lines of information requirements.
2.2.2 Derivation of Phonetic Segments

Allophone Selection and Coarticulation

The sequence of sounds submitted to the speech generator can be represented by a sequence of allophones, which is derived from a canonical phonetic transcription of the words in the utterance. This derivation should most notably account for certain so-called coarticulation effects, i.e., changes of the quality and quantity of phonetic segments in the presence of neighboring segments and due to their position within the utterance. Examples of such effects are:

- Unvoiced plosive consonants are more or less aspirated depending on their position and phonetic context. In the pair "stick"/"tickt" ("she"/"stitches"/"it"/"ticks"), phonetically [ʃtkt]/[tkt], the [t] sound at the beginning of [tkt] is much more aspirated than after [ʃ] in [ʃtkt] and than the final [t] in both words.

- The unstressed definite German article "der", in canonical phonetic form [deə], is usually reduced to [deə], or even to [deə] or [deə].

- The [n] in German "ein Bier" [ain biɐ] ("a beer") may be assimilated to the subsequent [b] sound in fluent speech, which results in the pronunciation [ain biɐ].

The number and degree of assimilations and reductions in an utterance depend on the precision and rapidity of articulation. How many allophonic variants and fine coarticulation effects can be and must be realized in a TTS system is primarily a question of the type of speech sound generator. Formant and articulator synthesis (cf. Chapter 6) allow a much finer control of sound qualities and transitions than concatenative synthesis.

Phonetic Transcription

The canonical phonetic transcription of a word, as given for German in [Dud74], is a standardized phonetic form which is more precise than a phonemic representation (for example, it distinguishes between two different allophones of the /r/ phoneme), but the inventory of allophonic variations is rather limited (for example, it does not distinguish different aspiration degrees of the [t] sound). The canonical phonetic transcription is determined by the phonemic representation of that word. For instance, the canonical German phonetic forms [biːɡ] and [biːʁ] ("Bier"/"beer" and "Biere"/"beers"), which are realizations of the phoneme sequences /biː/ and /biːʁ/, contain different allophonic versions of the same phoneme /r/. The allophonic variation in this example depends on whether a vowel or a consonant follows the /r/.

Phonemic Representation

The phonemic representation of a word is given by the concatenation of the phoneme sequences of the individual allomorphs of which the word is composed, where an allomorph is one particular grapheme and phonemic realization of a morpheme. For example, the German word <behandelt> ("treat", 1st person singular, present tense), consists of a prefix, a stem, and an inflection morpheme, which are graphemically and phonemically realized as the allomorphs <be>/bo/, <handel>/handel/, and <st>/st/.

The phonemic representation of the full word is therefore /behandelt/. Another inflection form of the same verb is <behandele> (2nd person singular, present tense), in which the allomorphs <be>/ba/, <handel>/handel/, and <st>/st/ yield the phoneme sequence /behandele/ for the whole word. In this example, <handel>/handel/ and <handel>/handel/ are allomorphs of the same verb stem morpheme, and the variation is expressed in both the graphemic and the phonemic realization. In other cases, however, the variation is only expressed in the phonemic realization of the allomorphs. For example, the realization of the noun stem "generator" ("generator") can be <generator>/generator/ or <generator>/generator/, depending upon whether the plural ending <en>/an/ follows the stem or not. (In English, the changes induced in the phonemic form of an allomorph (but not in the graphemic form) due to different morphological structures may even be more severe. Consider for instance English <photograph>/ˈfoʊtəɡrɑf/ versus <photograph>/ˈfɒtəɡrɑf/2, where the derivation suffix <y> changes the word stress position and thereby also all vowel qualities. In the similar German case <photograph>
/foʊˈɡraɪf/ versus <Photographie> /foʊɡraˈfiː/, only the length of the /a/ is changed.

Considering the dependencies stated above, the linguistically obvious way to obtain the phonetic transcription of a word is to morphologically decompose the word, to find the phonemic representation of all allomorphs in the word, and to convert the full phonemic form into the corresponding canonical phonetic representation by applying a set of pronunciation rules (for example, to generate the afore-mentioned two variants of the /r/ phoneme).

Allomorph Sequence

In reading aloud a written text, the only way to find out the sequence of allomorphs of a word is to look at the graphemic representation of that word and possibly of that of the surrounding text. In most cases in German or English, the graphemic representation of a single word is enough information to derive the corresponding phonetic transcription. For example, the English words <table> and <bite> and the German words <kind> and <gut> have a unique phonetic transcription, even though they may be ambiguous in their syntactic and semantic function. In other cases, more information is needed to find out the correct phonetic form of a word. Examples of such cases are:

- English <record> has different phonetic forms depending upon whether it is a verb or a noun (['rɪˈkɔːrd] vs. ['rekɔːrd]).
- German <modern> has the phonetic realizations ['moːdən] (verb) and [mɔˈdɛm] (adjective).
- The English verb <read> in the sentence “I read the book.” may be pronounced [rɛd] (present tense) or [rɛd] (past tense).
- In the German sentence “Ich werde den Zaun umfahren.” (“I will run down / drive around the fence.”) the word <umfahren> has the two possible pronunciations ['ʊmfaʁɐn] (“to run down”) and ['ʊmfaʁɐn] (“to drive around”), depending on the meaning of the sentence.

In the first two examples, syntactic information (i.e., information about the sentence structure) is sufficient to distinguish the variants. In the third case, the surrounding text material (if any) may provide the information about the correct tense, and in the fourth case, semantic information from the surrounding text must be considered in order to find out the correct pronunciation. If the third and fourth example are written in isolation, the ambiguity can simply not be resolved.³

Fortunately, ambiguities of the above type do not occur too often (in English they are probably more frequent than in German). However, it can be seen from the examples that for a TTS system to perfectly fulfill its task, a highly sophisticated analysis would be necessary even if only the pronunciation of a word should be obtained. For prosody information, the situation is even worse, as will be shown in Section 2.2.3.

Grapheme-to-Phoneme Conversion

From the linguistic view of the derivation of phonetic segments as sketched above it follows that a TTS system should first decompose words into morphs and find the corresponding phonemic representation. The application of several phoneme-to-alkaline conversion rules would then finally yield the representation of speech sounds to generate.

The decomposition of words into morphs with their graphemic and phonemic representation generally requires a complete lexicon of morphemes or allomorphs of a language. Such lexicons are, however, not ready at hand, and for a long time it did not appear feasible to incorporate such a lexicon in any TTS system at all.

TTS systems therefore developed in a different way: since in alphabetical languages (like the Indo-European languages) the graphemic representation of a word is in fact already a simplified (and sometimes outdated) phonemic representation, it is quite common for TTS systems to attempt to convert the graphemic representation of a word directly into the corresponding phonetic transcription by means of a set of so-called grapheme-to-phoneme conversion rules or letter-to-sound rules.

³In the written form, the third and fourth example contain both possible interpretations at the same time, and the ambiguity can even be transmitted to other persons by copying the sentence as written text. However, if the sentences must be read aloud or translated into another language, a decision must be made about the correct or plausible variant.
These “shortcut” rules are usually paired with a minimal morphological analysis during which prefixes, suffixes and inflection endings are stripped off the word. For frequent words that cannot be converted properly by this procedure, TTS systems typically comprise an exceptions dictionary which is consulted before the application of the regular letter-to-sound conversion procedure. Examples of letter-to-sound rules might be:

- The German <h> is converted into [h] if it is not located after <s>, <sc>, <t>, or <p>.
- The German <a> is converted into [a] if not preceded by <a> and if followed by <a> or <h>, or by only one consonant.

To establish a good rule-based grapheme-to-phoneme conversion procedure poses quite a lot of problems and requires highly sophisticated rule systems. The following examples should give an idea of these problems:

- The <e> in German <umgebucht> ("changed a reservation") is realized as [a], because it belongs to the past participle inflex <ge> [ge]. In <umgebung> however, <e> is pronounced as [ɛ] because it is the stressed vowel of the verb stem <geb>.
- The sequence <sch> in the German word <maschen> ("stitches") is pronounced as [ʃ], but in the German word <mäuschen> ("little mouse") it is pronounced as [sc]. In the first case, <sch> fully belongs to the noun stem <masche>, whereas, in the latter case, the stem <mäus> is followed by the suffix <chen>.
- Without knowing that <parabol> and <antenne> are (foreign) stems of the German language, it is hard for an automatic procedure to decide whether the word <parabolantenne> should be pronounced [parabolantenena] (for the correct stem combination “parabol-antenne”) or, for instance, [parabolantenena] (for the incorrect hypothetical decomposition “parabol-an-tenne”). In the latter example <antenne> is in fact a German noun stem so that even a system with a limited lexicon of German stems might decompose and pronounce the word incorrectly.

- In the two English words <photograph> and <photography>, as already mentioned in a preceding paragraph, the presence of the suffix <y> changes the quality of the first <o> in the word. A rule for the correct conversion of the <o> would have to take into account an extremely large graphemic context.

Some of these problems may seem far-fetched, but as a human being who knows the words of a specific language, one is often not aware of the problems and ambiguities that exist for an automatic letter-to-sound conversion.

With the increasing memory capacities of modern computers, new TTS systems clearly tend towards the use of large lexicons (of tens of thousands of allomorph or full-form entries). Besides being linguistically more adequate, the lexicon-based derivation of phonetic segments is much simpler to understand and realize than letter-to-sound rules. Nevertheless, as shown in [CC190], even extremely large lexicons never cover all words of a language and especially not all proper names. For these exceptional cases, lexicon-based TTS systems must still incorporate some sort of grapheme-to-phoneme conversion. Considering the fact that human beings possess an internal representation of most of the words and morphemes of a language, and that they are also capable of consistently pronouncing novel or nonsense words based on the graphemic representation (e.g., pseudo-German “zwil”, with the pronunciation [zwil]), the combination of large lexicons for the most frequent words and rule-based grapheme-to-phoneme conversion for exceptional cases seems linguistically quite appropriate.

The problem of mapping a grapheme sequence onto a phonetic transcription, and the solution adopted in the case of the SVOX system is further discussed in Chapters 3 and 4.

2.2.3 Derivation of Prosody

The physical prosodic parameters by which speech segments can be modified are fundamental frequency ($F_0$) or pitch, segment duration, and signal intensity. These measurable quantities are the physical correlates of the much more abstract linguistic notions of sentence melody, speech rhythm, and loudness. It is generally acknowledged that funda-
mental frequency and segmental duration are more important for the naturality of synthetic speech than intensity. Consequently, prosody research has concentrated much more on the first two parameters, and most current TTS systems do not include any explicit intensity control. However, if pitch and duration control were nearly perfect in a TTS system, a very good intensity control would become necessary as well, in order for the system to achieve complete naturality. For the time being, however, much more remains to be done in terms of the quality of melody and rhythm of synthetic speech.

Influences on the Physical Prosodic Parameters

The prosodic parameters needed to drive the speech generation component (i.e., fundamental frequency $F_0$, duration, and intensity) are influenced by several distinct factors:

- The prosodic parameters strongly depend on the speaker’s physiology and mental state; for example, women in general speak at a higher pitch than men do, and the speed and loudness of speech production rises with increasing excitement.

- The prosodic parameters depend on the speech segments (allophones) that are uttered; for example, vowels are generally uttered at higher signal intensity than consonants, the duration of sustainable sounds can be stretched much more than the explosion phase of a plosive consonant, and vowels articulated with a high tongue position tend to be realized at a higher pitch than others.

- There are some universal phonetic properties associated with prosody. One such phenomenon is the so-called declination of $F_0$: within each utterance there is a general tendency for $F_0$ to decrease from the beginning towards the end [CCT82]. This behavior seems to be physiologically motivated since it is related to the decreasing subglottal air pressure as air is pushed out of the lungs. However, the effect is of linguistic nature in the sense that it occurs in speaking while it is no problem to keep the pitch at a constant level in singing. The general behavior of the speech signal intensity within an utterance is quite analogous to that of $F_0$.

- In tone languages, such as Chinese, Thai, or Swedish, the type of $F_0$ movement within certain syllables may distinguish different meanings of a word. These tonal movement types are therefore of phonemic nature, but in TTS synthesis, of course, they must be treated in close conjunction with other melodic contributions rather than with the sound segment production.

- The prosodic parameters depend on the morphological, syntactic, semantic, and pragmatic composition of each utterance; for example, in the sentence “I have a car”, the word “I” is generally less prominent than the word “have” and is therefore realized more rapidly, with less intensity, and at lower pitch, unless it should be pointed out that it is not another person that has the car, which is indicated by strongly accenting the word “I”.

The influences of the first type, the so-called para- or extralinguistic factors, mostly characterize one particular speaker and a specific speaking situation. For the sake of simplicity, most TTS systems concentrate on modeling only a very limited number of speakers and speaking styles. In the SVOX system presented in this report, the aim was to keep these influences as constant as possible and to model only one single speaker and only one speaking style, namely a “neutral” information communication style, as applied by e.g., news readers.

The influences of the other types, the linguistic factors, vary with each utterance, so that they must be treated explicitly in a TTS system in order to achieve a natural-sounding synthetic speech signal.

Accentuation and Prosodic Phrasing

Besides universal properties (like the declination behavior of $F_0$), the physical prosodic parameters within an utterance strongly depend on the prominence of certain syllables over others and on the grouping of certain words into rhythmic and melodic units. In the German and English linguistic literature, these influences are termed accentuation and (prosodic) phrasing. The most obvious relationships between accentuation and phrasing and the prosodic parameters are:

- Strong accents are marked by a major rise or fall of $F_0$. 
• Accented syllables are longer than unaccented syllables. Either the syllable nucleus (vowel or diphthong) or the following consonant (after a short vowel) is lengthened.

• Signal intensity is increased within accented syllables.

• Syllables preceding a phrase boundary (or at the end of an utterance) are lengthened, and pauses may be inserted at phrase boundaries.

• Phrase boundaries may be indicated by a special melodic pattern (for example a fall-rise pattern in English), and resets of the \( F_0 \) declination may occur at phrase boundaries.

Morphological Information Needed for Prosody

Morphological information is needed for two important basic requirements in the derivation of prosody: syllable boundaries and word-stress positions.

The syllable is one of the most important units for prosody. The splitting of a word into speech syllables is partly defined by the morphological structure of a word (at least in the German language), partly by segmental phonetic criteria. Before German prefixes and stems, there is a mandatory syllable boundary. Before suffixes and inflection endings, on the other hand, syllable boundaries are set according to phonetic criteria. For example, the German word “versagen” (“to fail”) can be realized in the inflected forms \(<\text{versagen}>\) [\(\text{fäg-zag-\text{g}}\)], \(<\text{versagst}>\) [\(\text{fäg-zagkt}\)], and \(<\text{versagte}>\) [\(\text{fäg-zagkt-t}\)]. The first syllable boundary is determined as invariably lying between the prefix \(<\text{ver}>\) and the stem \(<\text{sag}>\), but the second syllable boundary (if any) is set such that exactly one consonant precedes the next syllable nucleus.

The word-stress positions denote syllables carrying the primary, secondary and possibly tertiary stress within a word. In German and other languages, these positions are the basis for sentence accentuation, which describes the relative prominence of the syllables of a full utterance. The word-stress position of a German word strongly relates to the morphological structure of that word. For example, the word “getragen” (“carried”) is phonetically realized as [\(\text{ga}^\text{t}ra\text{\textg}\)], with the (only) word-stress positioned on the stem \(<\text{trag}>\). In the word \(<\text{straßenverkehr}>\) \([\text{'\text{s}\text{tra}\text{\texts}\text{n}ver\text{\textk}\text{\textr}}\text{\textg}]\) (“street traffic”), the primary stress lies on the first stem \(<\text{strasse}>\), and the secondary stress on the second stem \(<\text{kehr}>\). The notions “primary” and “secondary” stress denote the levels of prominence of the stressed syllables of a word when this word is uttered in isolation. In combination with other words, the absolute levels of prominence of the same stressed syllables may be quite different. In some cases, the secondary stress of a word may even be realized more strongly than the primary stress, due to a rhythmical stress shift phenomenon. In this sense, word stress positions encode morphological information rather than prosodic prominence.

Syllable boundaries and word stress information are usually encoded in close conjunction with the segmental phonetic information in the phonetic transcription of a word. as for instance in [\(\text{fäg-\text{ke}s-\text{am-p}]\) \(<\text{verkehrsanspiel}>\) (“traffic light”), where ‘-’ denotes syllable boundaries and ‘、“ and ‘,‘ denote the primary and the secondary word stress.

Syntax, Semantics, and Pragmatics

Accentuation and Phrasing of an utterance are primarily defined by its semantic contents, in the sense that semantically more prominent items of an utterance are given more accentual weight, and that words that belong together semantically tend to be spoken in one rhythmic and melodic group. For example, in the sentence “I gave the new book to Mary,” if it is spoken as a “neutral” utterance, the words “book” and “Mary” are the most prominent items, and “the new book” and “to Mary”, which are semantic units, are also spoken as rhythmic units. If, however, the fact should be pointed out that it is not the old but the new book that was given to Mary, a contrastive, strong accent is put on the word “new”. In the utterance “Charles the first king of England”, the semantic contents define whether the correct grouping is “Charles // the first king of England” or “Charles the first // king of England”.

In addition to semantics, pragmatic features may highly influence the prosody of an utterance. For example, the order “give me the book, please” would be repeated with increasing strength and at an increasing pitch level, to indicate more and more forcefulness so long as the other person refused to hand the book over.

From the above considerations, a semantic and pragmatic analysis
of a text seems necessary for the correct derivation of the prosody of an utterance within that text. However, a TTS system can to some extent circumvent these analyses: Firstly, in most TTS systems, the pragmatic environment is constantly set to a “neutral” news communication style. Therefore, it is not necessary to analyze the pragmatics for each new utterance. (In the synthesis component of dialog systems, however, the need for pragmatically appropriate prosody is much stronger, in order, for instance, to correctly differentiate between questions, requests of different force, simple statements, etc.) Secondly, the semantics of an utterance is partly reflected in the syntactic structure, so that at least some prosody information can be extracted from the syntactic structure of an utterance alone. Examples of such semantics/syntax correlations are:

- Semantic units are often realized as syntactic units as well. For instance, “the beautiful new house” denotes a semantic object with certain specific properties, and this is expressed syntactically as one noun phrase. (In other cases, however, semantic units may be syntactically discontinuous, as in the case of the stressed German verb prefixes: in the German sentence “Ich schaue ihm das Buch zurück.” (“I will return him the book.”), the full verb consists of the parts “gebe” and “zurück”.)

- German nouns are generally more prominent than modifying adjectives or so-called function words (articles, prepositions, auxiliary verbs etc.). These distinctions can quite easily be obtained from a lexical and syntactic analysis.

- Persons or objects that are referred to several times in a text are less prominent than newly introduced items. This is expressed syntactically by replacing nouns and proper names by a personal pronoun or by applying the definite article to known objects and the indefinite article to newly introduced ones.

- Rare syntactic constructions may indicate specially focused items. For example, in the German sentence “ein Buch hat er dem Mann gegeben”, the use of an indefinite article in an object in accusative case at the head of the sentence indicates that this object receives special semantic weight and must be strongly accented.

- One of the nicest reflections of semantic focus in a syntactic construct is the French mise en relief: in the sentence “C’est lui qui a cassé le verre.” (“He has broken the glass”), it is clearly indicated that the focus of the utterance lies on the subject (“lui”), whereas the neutral variant of the same sentence would be “Il a cassé le verre.”. (It is interesting to note that, unlike in German or English, the focus cannot be put on the the word “il” by stronger accentuation. The change of the syntactic construct is mandatory in this case.)

### Punctuation Marks

A word should be said here about the utilization of punctuation information. Many TTS systems use punctuation information for the generation of prosody, since punctuation marks often give hints about where a human speaker would put phrase breaks and pauses. However, this information is in most cases (at least in German) defined on syntactic grounds. For example, in the German sentence “Ich schaue diesen Film, weil er so spannend war, zweimal an.” (“I saw this movie twice because it was so thrilling.”), the two commas are simply set around the intervening subordinate clause. In this case, the sentence structure could be analyzed correctly without considering the punctuation marks (as is actually the general custom in the field of natural language processing), and the commas are completely redundant. In most cases, punctuation mainly serves as a help for the reader to more easily and more rapidly process the written text.

Nevertheless, in some cases, written sentences may be ambiguous (which in many cases would result in different possible prosodic realizations), and punctuation marks can then indicate the correct meaning and syntactic structure of a sentence. For example, the German word sequence “die neue elektrische Schreibmaschine” (“the new electric typewriter”) has two different interpretations depending upon whether the new typewriter is the first electric one (“die neue elektrische Schreibmaschine”) or whether it is the latest of a series of electric typewriters (“die neue elektrische Schreibmaschine”). In this case, the presence or absence of the comma indicates the correct semantic variant.

Thus, from the point of view of linguistics, punctuation should be considered in a TTS system only for the resolution of real syntactic and semantic ambiguities. However, as long as a TTS system does not com-
prise an excellent syntactic and semantic analysis, punctuation marks may help as a guideline for the proper segmentation of sentences into speech groups. For this reason, punctuation is also currently utilized in the SVOX system presented in this report (cf. Chapter 5).

2.3 The Development of a TTS Model

A TTS system maps a linear sequence of elements of a graphemic medium onto a linear sequence of elements in an acoustic medium. According to the considerations in Sections 2.2.2 and 2.2.3, this mapping comprises two parts: the analysis of the graphemic string, during which an underlying representation of the original symbol sequence is constructed, and the subsequent use of this representation for the synthesis of the target sound sequence [HHP+87]. The TTS process can therefore be viewed as a text analysis stage followed by a signal synthesis stage, as depicted in Figure 2.1. (This model does not, of course, imply that a text or sentence must be fully analyzed before proceeding with the synthesis. The two processes may well be intermingled, as in the case of a human being reading aloud a certain text. In the SVOX system, however, the input text is processed sentence-wise.)

The underlying structure, which is the result of the text analysis, is usually some more or less detailed representation of the syntactic sentence structure, but it might well contain some further (e.g., semantic) information.\(^4\)

\(^4\)Concept-to-speech systems, as said in Section 1.2, do not need the text analysis component of a TTS system. Instead, a CTS system should incorporate a component that directly generates the underlying structure of the next utterance to be synthesized and submit it to the synthesis stage.

It is quite clear that the input text submitted to a TTS system does not contain any characterization of a certain (human or synthetic) speaker. On the other hand, the resulting speech signal will clearly identify a particular speaker speaking in a specific style. Thus, somewhere in the TTS synthesis process, there must be a transition from speaker-independent to speaker-dependent (or speaker-characterizing) information. Theoretically, this transition could be spread out over the entire TTS process. However, in the linguistic view adopted as a basis for the ETH TTS project, a speaker-independent part is strictly separated from a speaker-dependent part. In this model, which is shown in Figure 2.2, the terms transcription and phono-acoustical model have been adopted for the two basic TTS stages. The transcription maps (or transcribes) the graphemic text onto an abstract intermediate representation of the utterance to be synthesized, and the phono-acoustical model implements the phonetic-acoustic realization of this intermediate representation.

The intermediate representation is a phonological representation, which means that it consists of a minimal, speaker-independent description of the speech sounds and the prosody of the utterance to be
synthesized. This description is, like the input sentence, a linear sequence of symbols (cf. Chapter 5). Most TTS systems generate and collect intermediate phonological information in this sense, but hardly ever is it compiled and restricted in such an explicit way as in the SVOX system. The main reason for this explicitness was the linguistically motivated strong separation of transcription and phono-acoustical model, which was also very helpful in parallelizing the research work.

It should be pointed out here that the above model of a TTS system is very similar to the front end / back end model of computer program compilers as described in [ASU86]. There, the front end is the machine-independent part of the compiler, which analyzes a computer program and generates an intermediate program code. The compiler back end interprets the intermediate code and generates the (possibly optimized) machine code for a particular computer. This similarity between a TTS model and a compiler model is not too surprising if one considers the fact that both systems are language processing systems. The input of both systems (a natural language text or a computer program text) may even be analyzed by applying the same basic formalisms of grammars and lexicons, due to the common background of formal language theory [Cho57]. The difference in the syntactic analysis lies in the fact that computer languages usually belong to a simpler, more restricted language category than natural languages. (Computer-oriented artificial languages are often designed as context-free languages of LI-LI type, which means that they can be parsed unambiguously from left to right with only one symbol look-ahead. For parsing natural languages, on the other hand, much wider contexts must be taken into account, and very often ambiguities remain after parsing which could only be resolved by a semantic analysis. Moreover, there are some reported cases in which natural languages cannot be described by context-free grammars.)

2.4 The SVOX Architecture

2.4.1 Design Considerations

From the linguistic considerations in Sections 2.2.2 and 2.2.3 it is clear that the realization of a TTS system of very high intelligibility and naturalness would require very far reaching linguistic analyses. For feasibility reasons, TTS projects must decide how far they can go with a limited amount of computer power, linguistic knowledge, and research. This decision is made harder by the fact that, with a rather limited effort, it is possible to make even a small computer talk in a more or less intelligible way, while the higher the quality of a TTS system, the larger are the efforts needed for even small improvements. The question therefore remains how good a TTS application really needs to be and how much linguistic beauty and completeness should be incorporated in the system.

In order to simplify the general task of synthesizing any given text, the following basic aims and restrictions were envisaged at the start of the ETH TTS project and still hold for the SVOX system:

- The TTS system should be realized using the diphone synthesis developed in [Kae83] (cf. Chapter 6).

- The TTS system should be restricted to the synthesis of texts in the pragmatic context of "neutral" information transmission (as, for instance, reading news or weather forecasts), and the phon-acoustical model should imitate one specific speaker as closely as possible (the same speaker from whose voice the diphones were extracted), instead of allowing for many different voices or speaking styles.

- Semantic and discourse analysis was judged infeasible for a TTS project at that time, but a full lexicon-based morphological and syntactic analysis of German words and sentences was envisaged according to the tradition in the field of natural language processing.

- The phonetic transcription of words should be obtained merely by looking up full-forms or morphemes in lexicons containing the graphemic and phonemic representation for each entry. (This aim was partly revised for the SVOX system, cf. Chapter 4).

- Input texts should be analyzed sentence by sentence, since further context analysis was excluded for feasibility reasons.

- Since semantic information would not be available, the generation of prosody should be based mainly on the syntactic structure of
2.4.2 Structure of the SVOX System

The general TTS requirements discussed in Sections 2.2.2 and 2.2.3 and the special design restrictions presented in the previous section led to the overall TTS structure depicted in Figure 2.3. The figure displays the main processing modules and the information flow between them.

This flow can be summarized as follows: The input text is treated sentence by sentence. Each word in the sentence is looked up in a full-form lexicon or is morphologically decomposed into morphemes (according to a word grammar) which are looked up in a morpheme lexicon, and the syntactic structure of the whole sentence is subsequently analyzed according to a sentence grammar. The result of this analysis is an annotated syntax tree (i.e., a parse tree), which contains the structure of the whole sentence and the structure of each word. Moreover, for each constituent of the syntax tree, the constituent type and further attributes (like, e.g., case, number, and gender) are marked, and the graphemic and phonemic representation of words and morphemes are contained in the tree.

From the syntax tree, accentuation and prosodic phrasing are determined, which, together with the phonetic transcription of each word, constitute the phonological representation of the sentence. This representation is the input to the subsequent generation of duration values for all speech segments and of the fundamental frequency contour.

The speech signal is generated by concatenation of diphone elements (that is, by concatenation of speech sound transition elements extracted from natural speech). The diphones are selected according to the phonetic transcription of the utterance, and upon concatenation they are modified such that they match the specified duration and F0 values.

Whereas Figure 2.3 is a view of the general flow of information in the SVOX TTS process, Figure 2.4 is more precise in that it shows the sequential processing of a sentence when synthesized by the SVOX system. The figure also displays the chapters in which the individual
modules are explained. Going beyond Figure 2.3, two new modules appear here: the syllabification process, which is inserted after the syntax analysis, and a module which implements some (very few) regular allophonic variations (which could roughly be termed "coarticulation effects") at the transition point from phonology to the phonetic realization of an utterance.

2.4.3 Types of Processing

Transcription

The transcription part of the TTS system consists of merely symbolic computations, i.e., it maps symbolic data onto other symbolic data. Throughout the transcription, knowledge-based and rule-based subsystems have therefore been realized, and the aim was to strictly separate explicit, static, language-specific linguistic knowledge from the application of this knowledge.

As a matter of fact, it turned out in the SVOX system that this aim has been reached in the syntactic and morphological analysis (including even grapheme-to-phoneme mapping), as will be shown in Chapters 3 and 4: All syntactic and morphological analyses are bidirectional, i.e., the system is able to perform text-to-phoneme mapping as well as phoneme-to-text mapping without changing any data or algorithms. (In order to turn around the processing direction it is only necessary to exchange the location of graphemic and phonemic data. For demonstration purposes, this can be done in the SVOX system by switching a single system flag.) The ability to change the direction of processing does not mean that the SVOX system could easily be turned into a speech recognition system, but it nicely shows the high degree of explicitness of linguistic knowledge in this part of the system.

The remainder of the transcription (accentuation and prosodic phrasing) is much more processing- and algorithm-oriented and has a specific direction of operation. Future research might, however, attempt to make these components as general as the morpho-syntactic analysis.

---

Figure 2-4: The sentence processing stages in the SVOX system. The individual modules are further explained in the chapters indicated in the figure.
Phono-Acoustical Model

The phono-acoustical model maps symbolic data onto acoustic, numerical data. This means that at some point in this mapping statistical data must be handled. It was not clear at the beginning of the TTS project how far symbolic (rule-based) computation should go into this phonetic realization stage and at which point numerical parameters should enter. However, after trying different approaches, it turned out that the most successful prosody control was achieved by applying very direct statistical mappings from the phonological data onto the final numerical parameters: The diphone synthesis approach consists of picking typical representative speech samples out of a multitude of possible realizations, and thereby directly maps the phonetic symbols onto the speech signal. Duration and $F_0$ control were realized by applying trainable statistical models (a generalized linear model and an artificial neural network), which directly map the phonological symbols onto phone duration and $F_0$ values. Thus, in very strong contrast to the transcription, the phono-acoustical model only incorporates implicit, statistical phonetic knowledge. Whereas the SVOX transcription contains more explicit knowledge than other TTS systems, the phono-acoustical model of the SVOX system incorporates implicit knowledge to a much higher degree than comparable components of many other traditional TTS systems.

2.4.4 Implementation Overview

This section presents a short general overview of the SVOX implementation. A much more detailed description is given in [Tra95c].

The current SVOX system has been implemented in Modula-2 [Wir88] and Prolog [CM84, Mul85, Tra95a], and some time-critical routines were realized in VAX/VMS assembler [DEC90]. The syntactic and morphological analysis has been written entirely in Modula-2. The modules in the remainder of the system are used via Prolog interfaces. Some of these modules have been written exclusively in Prolog. Other modules have been written in Modula-2 or assembler, but their main routines are also accessible in Prolog programs.

The main functionality of the SVOX system is quite easily portable due to the fact that the Prolog interpreter system incorporated in the SVOX system was itself written in Modula-2, and that for all assembler routines there exist corresponding pure Modula-2 procedures. SVOX currently runs on VAX/VMS, Sun SparcStation/UNIX, and Macintosh.

The choice of Prolog as a major implementation and interface language was motivated by the numerous symbolic computations in the synthesis process, by the suitability of Prolog as a flexible, interpreted experimental environment language, and, of course, by the personal taste of the researchers. In order to somewhat overcome the slowness of the Prolog interpreter, the Prolog code can be compiled into machine code on VAX/VMS [Tra95a]. However, for a really fast and small implementation, the Prolog parts of the system would have to be rewritten in another programming language. But with the ever increasing power of modern computers, a real-time system may be most easily obtained by simply waiting some two or three years.5

2.5 Other TTS Architectures

Most existing TTS systems are not much different from each other as far as the overall architecture is concerned [ e.g., MITalk [AHK87], Infovox [LLG93], CNETVOX [LEM89], or the BT system [Gav93]]. The individual subparts, however, may vary substantially from one implementation to the other. For example, in the phono-acoustical model in the SVOX system, statistical approaches are applied that incorporate mainly implicit phonetic knowledge, whereas in many other systems (e. g., MITalk or Infovox), the phonetic realization of speech is treated in a much more explicit, rule-based manner.

---

5In the recent diploma thesis [Vet95] a large part of the original Prolog synthesis procedures (except for duration control) were rewritten in Modula-2, which considerably increased the speed of the system and reduced its memory requirements.
Chapter 3

Syntactic and Morphological Analysis

3.1 Syntactic and Morphological Analysis in a TTS System

Syntax analysis in a TTS system serves several purposes:

- The syntactic structure of a sentence is determined. This information is highly relevant for the derivation of accentuation and prosodic phrasing of the sentence.

- Many ambiguities between homographs (i.e., words with identical spelling but different meaning and/or pronunciation) are resolved by a syntactic analysis. For example,

  German “modern” [ˈmoʊdən] (verb) vs. “modern” [ˈmoʊdən] (adjective)

  German “der kommt nicht mehr” vs. “der Mann kommt nicht mehr”, where “der” in the first case is a strongly accented demonstrative pronoun, whereas the second “der” is a completely unaccented definite article.
German "ich gebe zu1, zu2 den andern zu3 gehören"
In this example, "zu1" is a stressed verb prefix, which carries a strong accent, "zu2" is a preposition which remains unstressed, and "zu3" is an infinitive particle, which is unstressed, and which may even be more reduced than the preposition.

- Syntax analysis may help to determine the proper pronunciation of abbreviations and ordinal numbers. For example
  "am 11. Oktober" ("am elften Oktober", dative case)
  "der 11. Oktober" ("der elfte Oktober, nominative case)

  "Meier u. a." ("Meier und andere", nominative case)
  "bei Meier u. a." ("bei Meier und anderen", dative case)

- Syntax analysis serves as a basis for the semantic analysis of sentences, which will be attempted in future TTS systems.

In order to carry out a syntactic analysis of a sentence, the syntactic function of each word in that sentence must be known. The most simple means of accessing this information would be a so-called full-form lexicon, i.e., a lexicon in which complete word forms are stored in their orthographic form together with some attributes describing their syntactic categories and further properties (e.g., case, number, and gender information and the phonetic transcription). However, in languages like German, which make use of a great number of inflected forms (verbs, nouns, adjectives) and compound words (mostly nouns in German), a full-form lexicon alone does not seem appropriate. Structured words should rather be morphologically decomposed into individual morphs with the help of a morph or morpheme lexicon.

The purposes of a morphological analysis of words are therefore:

- to find out the structure of compounds and inflected forms in order to derive the correct word category and other syntactic attributes (case, number, gender, person, etc.)
- to obtain the proper pronunciation of the whole word from the pronunciation of the morphs of which the word is composed.

In general, a morphological analysis yields several different possible readings (i.e., ambiguous analyses) of a word. For example, the German word "geht" can be 3rd person singular or 2nd person plural of the verb "gehen". The correct form must be chosen by the syntactic analysis or, if any, by the semantic analysis of the sentence.

Most TTS systems carry out a syntactic and morphological analysis to some degree, but the methods applied and the completeness of the analysis vary considerably. In the ETH TTS project and the SVOX system, a quite general and uniform approach was pursued, which will be explained in the next sections.

3.2 Morpho-Syntactic Analysis in the SVOX System

3.2.1 Architecture Overview

The syntactic and morphological analysis was designed and first implemented by Thomas Russi [Rus90, Rus91, Rus92]. Its original architecture is depicted in Figure 3.1. Modifications to this architecture will be described in 3.2.3. (Morphological and syntactic analysis together will frequently be termed "morpho-syntactic analysis" hereafter.)

Word and Sentence Analysis

The input text in the morpho-syntactic analysis is treated sentence by sentence. Each sentence is split up into orthographic words, and each word is submitted to the word analysis. The word analysis looks up words in a full-form lexicon, which contains full word forms in their graphemic and phonemic representation. Moreover, the word analysis attempts to analyze the morphological structure of the words. To this end, all morphemes occurring in a word are collected by looking them up in a morpheme lexicon and all morphologically meaningful morpheme sequences that span the entire word are sorted out by a word grammar.

The morpheme lexicon, like the full-form lexicon, contains entries in their graphemic and phonemic representation, and both lexicons con-
mORPHO-SYNTACTIC ANALYSIS IN THE SVOX SYSTEM

3.2. Morpho-Syntactic Analysis in the SVOX System

morphology of Koskenniemi [Kos83], mediate between the graphemic or phonemic surface representation of words (i.e., the regular orthographic or standard phonetic form) and a lexical representation of the word or its morphemes. The two-level rules are regular transformations between two levels, the surface and the lexical level.

Examples of phenomena that are handled by these rules in the current SVOX system are (‘+’ denotes the morpheme boundary between stem and inflection ending):

- the deletion of the <e> (and, accordingly, the /a/ phoneme) in some German verb stems before an inflection ending that starts with <e> (or /a/, respectively), as in
  \[\text{<hande> /handə/ (verb stem)}\]
  \[\text{<handel+st> /handel+st/ (2nd person singular)}\]
  but \text{<handl+e> /handl+ə/ (1st person singular)}

- the deletion of the <s> /s/ of the verb ending of the 2nd person singular after a verb stem ending in the phoneme /s/, i.e., in the grapheme <s>, <z>, or <x>, as in
  \[\text{<sitz+t> /sɪtz+t/}\]
  instead of the regular form
  \[\text{<sitz+st> /sɪtz+st/}\]

- the devoicing of stem-final plosives and fricatives in word-final position or before consonant (the German “Auslautverhärtung”), as in
  \[\text{<mond+e> /mønd+ə/ (plural)}\]
  but \text{<mond> /mønt/ (singular)}

By virtue of the two-level rules, the morpheme lexicon in the morpho-syntactic analysis was partly realized as a lexicon actually containing morpheme entries rather than allomorph entries. For example, the verb stem <hande> /handə/ was stored in this basic representation only, and not in the form of its two allomorphs <handl> /handl/ and <handel> /handel/. However, in other cases, for instance for the noun stem unumlaut in plural forms (as in singular <baum> /baum/ vs.
plural \(<\text{bäum}\> /\text{baym/})
the two allomorphs were stored in the lexicon.
This phenomenon could in principle be treated by two-level rules as well.
but the solution using two allomorphs seemed simpler and more appropriate in this case.
Thus, the “morpheme” lexicon was in fact realized as a mixture of a morpheme and an allomorph lexicon.

### 3.2.2 Original Realization

In the ETH TTS Project, the morpho-syntactic analysis was originally implemented in LISP [Rus90].
The grammars and lexicons which are still used in the present SVOX system, were realized by Hans Huenker [Huc90].
The two-level rules were established by Ruth Rothenberger [Rot91].
Some extensions to the two-level rules were added by the author.

Grammars were written in the UTN (unification-based transition network) formalism [Rus90], which is a modification of the well-known ATN (augmented transition network) formalism [Woo70].
UTNs, like ATNs, are capable of describing context-sensitive languages.
UTN grammars and lexicon entries were specified in the form of LISP terms.

The original morpho-syntactic analysis was a stand-alone program package.
which comprised a very flexible and general chart-parsing environment (see Section 3.4.3) for different parsing strategies and unification methods.

### 3.2.3 Modifications in the Morpho-Syntactic Analysis

As already mentioned in Chapters 1 and 2, a re-implementation of the morphological and syntactic analysis became necessary primarily due to problems that occurred when the grapheme-to-phoneme conversion presented in Chapter 4 was realized.

In this re-implementation, the original overall design was fully preserved and even slightly enlarged in the sense that two-level rules are now also applied between the surface and the lexical level of the full-form lexicon, rather than for the morpheme lexicon only as shown in

Figure 3.1. The main motivation for this generalization was the inclusion of two-level rules that handle the equivalence of the German umlaut characters “ä”, “ö”, “ü” and their substitutes “æ”, “œ”, “œ”, such that, e.g., “fuer” and “fuir” are both mapped onto the single lexicon entry \(<\text{fü}/>.
(In the weather forecast texts used in the prototype application of the SVOX system, the umlaut characters are always replaced by their substitutes. By extending the application of two-level rules from the use with the morpheme lexicon only to the full-form lexicon, these substitutions could be handled without including any new lexicon entries.)

Apart from the afore-mentioned slight modification of the system design, quite a few changes in the grammar and lexicon formalism and in the internal processing were realized in the course of the re-implementation of the morpho-syntactic analysis. The most important of these changes were:

- In order to be consistent with the remainder of the system and in order to easily allow the linking of the full TTS system into a single program, Modula-2 was chosen as the implementation language (instead of the former LISP).

- The grammar formalism was changed from the UTN formalism to the equally powerful but somewhat more concise DCG formalism (see Section 3.2.4).

- The lexicon formalism was simplified and rendered more TTS-specific, and a specialized lexicon-editing facility was added to the system.

- Although the chart parsing algorithm applied in the original implementation had a complexity of \(O(n^3)\), with \(n = \) “length of input sentence”, two specific algorithms of the morpho-syntactic analysis, the segmentation of words into morphemes and the building of parse trees, had a theoretical complexity of \(O(2^n)\).

These two algorithms were modified, such that in the SVOX system the entire morpho-syntactic analysis can now be done with a complexity of \(O(n^3)\).
3.2.4 Definite Clause Grammars (DCGs)

Definite Clause Grammars (DCGs; [PW80]) are extensions of context-free grammars. Context-free grammars are usually represented by context-free production rule systems, such as the Backus-Naur formalism (BNF). In DCGs, the extension from context-free to context-sensitive grammars is done by augmenting the context-free production rule skeleton with feature terms and the term unification operation. DCGs have been developed in close connection with the programming language Prolog (e.g., [CM84]), and it is in fact possible to automatically convert a DCG into a Prolog program that is a parser and at the same time a generator for the language defined by this DCG. The DCG-to-Prolog conversion is shown for example in [CM84, PS87].

Although it is very easy to build extremely flexible parsers and generators for DCGs in Prolog, the price of this flexibility is the often low efficiency of Prolog implementations and the exponential parsing complexity when the usual Prolog processing strategy is used. (This can, however, be somewhat counteracted by a careful design of the grammar, as shown in [PW80].)

In the SVOX system, DCGs are processed by an efficient chart parser, which was implemented in Modula-2. Moreover, the basic DCG formalism has been slightly enriched in order to select optimal solutions among several ambiguous solutions, and in order to control the building of parse trees.

The following sections describe the SVOX implementation of DCGs in more detail.

3.3 Formalism of the Morpho-Syntactic Analysis

The basic constructs in the morpho-syntactic analysis from the point of view of formalism are feature terms, lexicons, grammars, constituent declarations, and two-level finite state transducers.

The exact definition of the whole formalism is given in Appendix B.

3.3. Formalism of the Morpho-Syntactic Analysis

Throughout the formalism, reserved keywords start with `!`, identifiers are composed of letters, digits, and some special characters, strings are enclosed in double quote marks, and integer numbers obey the usual convention. Upper- and lowercase letters are not distinguished in identifiers. Keywords, however, must be written in uppercase letters.

Line comments (until the end of the line) start with `!` sign. and range comments start with `[` and end with `]`. Range comments may be nested.

3.3.1 Feature Terms

The SVOX DCG implementation uses feature terms and a term unification scheme which is closely related to the term unification used in Prolog. In SVOX, terms can be atoms, variables, or tuples.

Atoms are constants, denoted by identifiers, like

```
  nom noun_phrase x-y-z# ABC
```

Variables are denoted by a variable identifier, which starts with a `?` sign and continues like other identifiers. The variable `?` is the anonymous variable, like `.` in Prolog, which denotes a unique new variable every time it occurs. Anonymous variables cannot be referred to anywhere else and are usually applied as "don't-care" markers. Examples of variables are

```
  ?xyz  ?121  ?---  ?
```

The scope of a variable, i.e., the range in which equal variable identifiers denote identical variables, is one grammar rule or one lexicon entry (see below).

Tuples are the only means of building structured terms. They are simply lists of subterms, which are separated by commas and enclosed in parentheses. Each subterm is again a term. Examples of tuples are

```
  ()  (a, (b, c))  (a, (?x, n, (?y, (e1))) )
```

Tuples are mainly used to group features together. They correspond to Prolog structures with a functor and zero or more arguments. There is no equivalent to the Prolog list notation in SVOX
terms. Instead, varying-sized lists must be constructed explicitly by recursively using pairs of list elements and the remainder of the list. Thus, for example the term
\[(a, (b, (c, nil)))\]
could be used to denote a list of atoms ‘a’, ‘b’, and ‘c’. However, such lists have not been used so far in the morpho-syntactic analysis of SVOX.

### 3.3.2 Unification

The unification operation in the SVOX system is defined like the term unification in Prolog [CM84]. It tests whether two terms can be unified, and if so, it constructs the most general unified term corresponding to the two original terms. Two terms can be unified if and only if the two terms can be made identical by the substitution of variables by terms. For example, the two terms
\[(a, ?x, ?y, (b, (g, ?z)))\] and \[(?u, ?v, ?w, (b, (g, h)))\]
can be made identical by the substitutions
\[
?u \rightarrow a, ?x \rightarrow ?y, ?y \rightarrow ?v, ?z \rightarrow h
\]
A list of substitutions which unifies two terms is called a unifier of these terms. A common unified term is constructed by applying these substitutions (i.e., the unifier) to all variables of one of the original terms. With the above substitutions, the unified term becomes
\[(a, ?v, ?v, (b, (g, h)))\]

Usually, one is interested in the most general unified term, that is, the term obtained by as few substitutions as possible, as is the case in the above example. A less general unified term would have been
\[(a, (f, ()), (f, ()), (g), (b, (g, h)))\]

The unification operation tests the equivalence of values and binds variables to certain values. However, unlike in procedural programming languages, it lies in the nature of the unification operation that “testing operations” and “value assignments to variables” and unification among several terms can be done in any order. The reasons for this are that variables can only be bound once to a term and that two variables can be unified and thereby become identical (so-called “sharing” variables), which allows the delayed evaluation of the equivalence of atoms for example.

### 3.3.3 Grammars

Grammars are collections of grammar rules. A sample grammar is

\[
\begin{align*}
[R1] & \text{s} \quad \Rightarrow \text{np}(?c, ?n, ?q) \quad \text{vp}(?n) \quad \ast 10 \\
[R2] & \text{np}(?c, ?n, ?q) \quad \Rightarrow \text{opt\_art}(?c, ?n, ?q) \\
& \quad \text{opt\_adj}(?c, ?n, ?q) \\
& \quad \text{noun}(?c, ?n, ?q) \quad \ast 1 \\
[R3] & \text{opt\_art}(?c, ?n, ?q) \quad \Rightarrow \text{art}(?c, ?n, ?q) \quad \ast 1 \\
&R4 \text{ opt\_art}(?, ?, ?) \quad \Rightarrow \ast 1 \quad \text{INV} \\
&R5 \text{ opt\_adj}(?c, ?n, ?q) \quad \Rightarrow \text{adj}(?c, ?n, ?q) \quad \ast 1 \\
&R6 \text{ opt\_adj}(?, ?, ?) \quad \Rightarrow \ast \quad \text{INV} \\
[R7] & \text{vp}(?n) \quad \Rightarrow \text{verb}(?n) \quad \ast
\end{align*}
\]

This grammar states that a sentence (‘s’) is composed of a noun phrase (‘np’) and a verb phrase (‘vp’), which must agree in the number feature, and that the noun phrase consists of an optional article (‘opt\_art’), an optional adjective (‘opt\_adj’), and a noun (‘n noun’), which must agree in case, number, and gender.

Each grammar rule consists of a head, which denotes a constituent, and, after the production sign, a body, which denotes a list of subconstituents. The body is terminated by the ‘*’ sign. Empty subconstituent lists denote the empty production (R4). Each constituent or subconstituent is composed of a constituent identifier and a list of feature terms associated with the constituent, which are separated by commas and enclosed in parentheses (in fact, the feature list can be regarded as one term tuple). Empty feature lists are denoted by ‘()’ or by completely omitting the feature list (as in the head of R1).

The head of the first rule in a grammar is the start symbol of this grammar.

Grammar rules can optionally be followed by an integer penalty value, which can be used to select the optimal solution out of a num-
ber of ambiguous solutions. Penalty values are usually positive, but it is possible to apply negative values as well. If the penalty value is omitted, the minimal penalty value of 1 is assumed.

The keyword ‘:INV’ may optionally be set after the penalty value or after the ‘*’ sign (R4 and R6). This *invisibility flag* declares a rule to be invisible that is, its application will not produce a node in the resulting syntax tree. Usually, only empty productions are declared invisible. However, if invisible rules do have subconstituents, these will simply occur as direct descendants of the next higher visible syntax tree node. (The invisibility flag was introduced originally for technical reasons in the automatic conversion of UTN grammars to DCGs. In this conversion, auxiliary constituents were created, which should not be visible in the syntax tree.)

The penalty values in the sample grammar given above merely serve to illustrate the syntactic position of such values within grammar rules. The following example shows how penalty values could be used to prefer the nominative case of a noun phrase over other case variants in ambiguous situations:

\[
\begin{align*}
np'(nominative,nom,sg3,nom,sg3) &\rightarrow np(nom,sg3) \cdot 1 \\
np'(nom,sg3,nom,sg3) &\rightarrow np(nom,sg3) \cdot 2 \\
np'(sg3,nom,sg3,nom) &\rightarrow opt\_art\_sg3(nom,sg3,nom,sg3) \\
nom\_adj\_sg3(nom,sg3,nom,sg3) &\rightarrow \text{opt}\_adj\_sg3(nom,sg3,nom) \\
nom\_sg3(nom,sg3,nom,sg3) &\rightarrow \text{nom}\_sg3(nom,sg3,nom) \\
\end{align*}
\]

In this example, the auxiliary constituent ‘np’ actually defines the syntactic form of a noun phrase, and by the two rules for the ‘np’ constituent different penalty values are assigned to the noun phrase, depending on its case.

The same effect could be obtained by a dedicated empty evaluation constituent, for example:

\[
\begin{align*}
np(nom,sg3,nom,sg3) &\rightarrow opt\_art\_sg3(nom,sg3,nom,sg3) \\
opt\_adj\_sg3(nom,sg3,nom,sg3) &\rightarrow \text{opt}\_adj\_sg3(nom,sg3,nom) \\
nom\_sg3(nom,sg3,nom,sg3) &\rightarrow \text{nom}\_sg3(nom,sg3,nom) \\
case\_eval\_sg3(nom) &\rightarrow \text{case}\_eval\_sg3\_nom(nom) \\
case\_eval\_sg3(nom,sg3,nom,sg3,sg3) &\rightarrow \text{case}\_eval\_sg3\_nom\_sg3(nom,sg3,nom,sg3,sg3) \\
\end{align*}
\]

The invisible auxiliary constituent ‘case\_eval’ in this example does not change the definition of the syntactic form of a noun phrase but merely serves as an evaluation function of the case feature of the noun phrase.

### 3.3. Formalism of the Morpho-Syntactic Analysis

The invisible auxiliary constituent ‘case\_eval’ in this example does not change the definition of the syntactic form of a noun phrase but merely serves as an evaluation function of the case feature of the noun phrase.

#### 3.3.4 Lexicons

Lexicons can be regarded as collections of grammar rules in which so-called *preterminal symbols* (i.e., word or morpheme categories) produce terminal symbols (i.e., the individual words or morphemes). Lexicon entries could therefore be represented by production rules, like other grammar rules. However, in the SVOX system, a TTS-specific form of the lexicon entry is applied, as shown in the sample lexicon:

\[
\begin{align*}
[L.1] \text{con} &\rightarrow \text{"und"} & \text{"<2nt"} \\
[L.2] \text{noun(nom,sg3,neu)} &\rightarrow \text{"kind"} & \text{"kint"} \\
[L.3] \text{noun(acc,sg3,neu)} &\rightarrow \text{"kind"} & \text{"kint"} & 10 \\
[L.4] \text{art(nom,sg3,neu)} &\rightarrow \text{"das"} & \text{"das"} \\
[L.5] \text{art(acc,sg3,neu)} &\rightarrow \text{"das"} & \text{"das"} \\
[L.6] \text{verb(sg3)} &\rightarrow \text{"spielt"} & \text{"Sp1\_lt"} \\
\end{align*}
\]

(L.3 could actually be interpreted as a production rule of the form

\[
\text{noun(acc,sg3,neu,\"kint\") \rightarrow \text{\"kind\"}} \cdot 10
\]

with ‘kind’ being a terminal symbol. In this form, the phonetic transcription would have to be specified as a feature term.)

Each lexicon entry consists of a *constituent* identifier and an optional *feature list*, followed by two strings which define the *graphemic* and the *phonemic representation* of the terminal element. The phonemic representation is given in an ASCII-coded form of the IPA alphabet as defined in Appendix A. Although it is possible to use variable names and structured terms in feature lists of lexicon entries, they will usually consist of atomic values only, like case, number, and gender information.

The graphemic and the phonemic string may be empty. All words with empty graphemic string (i.e., “empty words”) will automatically be inserted (i.e., hypothesized) in the parsing process at every possible position in the input.
3.3. Formalism of the Morpho-Syntactic Analysis

Of course, in this derivation process, the same definite clause may be applied several times with different variable substitutions, i.e., in different instances.

The definition of the semantics of DCGs, which belong to the class of context-sensitive grammars, shows that the context-free skeleton of a DCG in general describes a larger language than the DCG, since the additional unification operation can only restrict the application of productions, but not extend it. By this restriction, a DCG in general exhibits a higher degree of selectivity than context-free grammars. For example, it is possible with DCGs (as with any context-sensitive grammar formalism) to define a grammar that exactly accepts sentences of the form $a^n b^n c^n$, i.e., a series of equally long sequences of identical words, which is not possible with context-free grammars.

Taking the grammar rules of Section 3.3.3 and the lexicon of Section 3.3.4, the sentence “das Kind spielt” can be derived from the extended start symbols “s[ ]” as shown in Figure 3.2.

All steps in the derivation of a sentence from the start symbol of a grammar can simultaneously be represented in the concise form of a parse tree or syntax tree, as displayed in Figure 3.3. This representation no longer shows the (usually irrelevant) order of production rule applications.¹

Unlike with formal languages (such as computer programming languages), natural language parsing often leads to several ambiguous derivations and hence to different syntax trees. In the above example, the noun phrase “das Kind” could also have been interpreted as an object in accusative case instead of the subject in nominative case. In the SVOX implementation of DCGs, each grammar rule and each lexicon entry carries a penalty value, which is used to classify different parsing derivations by their quality. The syntax tree in Figure 3.3 displays the summed-up penalty value for each node. The penalty value of a node is the sum of the penalties of all its (visible and invisible) descendants plus the penalty value of the grammar rule corresponding to this node. Thus, a tree with a lower penalty value in its root node

¹The SVOX system on VAX/VMS is able to display syntax trees in a genuinely tree-like graphical form similar to Figure 3.3 and in a space-saving indentation form. Syntax trees in this report are also shown using these two forms of representation, and they logically correspond to the syntax trees generated by SVOX, but they are not fully identical with SVOX trees as far as their graphical appearance is concerned.
3.3. Formalism of the Morpho-Syntactic Analysis

Figure 3.2: A possible DCG derivation of the sentence “das Kind spielt” from the extended start symbol “s()”. Variables with identical name denote sharing variables: After the application of L4, all variables are bound to values given in the lexicon entry L4 and can no longer assume other values.

Figure 3.3: Syntax tree representation of a possible DCG derivation of the sentence “das Kind spielt” from the start symbol “s()”. For each node in the tree, the constituent and features are shown as well as the summed-up penalty value. The empty constituent “opt_adj” was declared invisible and does therefore not appear in the tree.
is judged better than one with a higher value. In the given example, the assumption of accusative case for the noun phrase would lead to a higher penalty value for the parse tree, and therefore the nominative case will be preferred.  

3.3.6 Constituent Declarations

Unlike in the original UTN formalism, which applied named features, the interpretation of feature terms in the SVOX DCG formalism is strongly position-dependent. In order to document the meaning of feature terms and in order to allow the checking of lexicon feature values, constituents may be declared. Such declarations are also a vital part of the lexicon-editing facility described in Section 3.5.

The declaration of constituents is optional. Even if constituents are not declared explicitly, the usage of features is checked in all SVOX grammars and lexicons, and an error message is generated whenever there is a mismatch in the number of features used in different occurrences of a specific constituent.

Examples of constituent declarations are:

[D1]:CONS prep *

[D2]:CONS possp :FEAT case number *

[D3]:CONS n "noun" :
 :FEAT ca "case" {nom "nominative" gen "genitive"
 dat "dative" acc "accusative"}
 nu "number" {sg pl}
 ge "gender" {fem mas neu}
 :AND <accent true> <accpat left>*

Constituent declarations for a certain constituent maximally define the names of all features of a constituent and all atomic values that each feature can assume, and also some additional constant feature name/value combinations. Constituents, feature, and value identifiers may be followed by a string which presents a colloquial interpretation of the identifier for documentation purposes and for use in the lexicon editor. All declarations are terminated by ‘*’. The minimal constituent declaration consists of the constituent identifier and the feature names, preceded by the keywords ‘:CONS’ and ‘:FEAT’, respectively. For constituents without any regular features, the keyword ‘:FEAT’ must not occur.

Constituent declarations can be used to declare some constant feature name/value pairs for the constituent as additional features. These features, which are listed after the keyword ‘:AND’, are not used in the parsing process and do not appear in the syntax tree. Instead, the values of the additional features of a constituent type are accessed by specific inquiry routines. Additional features may therefore serve as constituent-related information for TTS steps which are carried out after the morpho-syntactic analysis (e.g., accentuation and phrasing). For example, in D3 the additional feature ‘accent’ with the value ‘true’ declares the constituent ‘n’ to be accentable, and the feature ‘accpat’ declares the accentuation pattern for the constituent ‘n’ as ‘left’ (i.e., main accentuation on the leftmost part of a noun).  

Constituent declarations can occur anywhere in grammars and lexicons and in separate lexicon descriptor files, which are simply collections of comments and constituent declarations. If upon loading SVOX grammars or lexicons, constituents occurring in rules or lexicon entries do not match the corresponding constituent declaration, or if constituents are redeclared differently, error messages are generated. Moreover, if the value-checking mode is enabled (see [Tra95c]), the correctness of the values in the feature lists of lexicon entries is checked and an error message is generated when a feature value does not occur in the value list of the corresponding constituent declaration, if a value list has been declared at all.

\[<text>\]
3.3.7 Two-Level Rules

Two-level rules [Kos83, Rus90, Rus92] serve to map a surface graphemic or phonetic representation onto a lexical representation, and also vice versa. This is achieved by associating each surface character with a lexical character. The use of the empty character (henceforth denoted by '∅') is allowed on both levels. For example, the verb form <sitzt> (2nd person singular), in which an <s> has been deleted from the regular inflection ending <st>, can be mapped onto the lexicon form <sitz> + <st> by the associations

<table>
<thead>
<tr>
<th>surface</th>
<th>lexic</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>i</td>
</tr>
</tbody>
</table>

The possible surface-lexicon correspondences are defined in part by an alphabet of lexicon/surface character pairs. The character pairs in the above example must all belong to this alphabet in order for the mapping to apply. Moreover, the possible mappings are further restricted by the actual two-level rules, which define allowed sequences of character pairs. These rules are usually given in one of the three forms

\[ X : Y \Rightarrow L \_ R \quad X : Y \Leftrightarrow L \_ R \quad X : Y \Rightarrow L \_ R \]

In these forms, \( X \) denotes the lexicon character of a pair, \( Y \) denotes the corresponding surface character, \( L \) stands for the left and \( R \) for the right context, which are defined using regular expressions over the character pair alphabet, \( \_ \) denotes the position of the character pair in question. \( \Rightarrow \) means that the pair to the left implies the context given on the right hand side, i.e., the pair must not occur in any other context than the given one (context restriction). \( \Leftrightarrow \) means that the given contexts \( L \) and \( R \) and the presence of the lexical character \( X \) imply the mapping onto the surface character \( Y \), i.e., no other pair with lexicon character \( X \) may be applied in these contexts (surface coercion). \( \Rightarrow \) simply requires that both implications, \( \Rightarrow \) and \( \Leftrightarrow \), hold simultaneously.

For example, the two-level rule used to describe the \(<s>\)-deletion in the SVOX system is

\[ s:∅ \Leftrightarrow (s|z|x) \_ :∅ +=:∅ \_ t \]

This rule states that \(<s>\) before \(<t>\) is deleted on the surface after an A-type verb stem (a present-tense verb stem) which ends in \(<s>\), \(<z>\), or \(<x>\). (Character pairs with identical lexicon and surface character are denoted by one character only.) The lexicon form of present-tense or completely regular verb stems (as, e.g., "sitz") in the SVOX morpheme lexicon is \(<sitzA+>\), where 'A' indicates the present-tense verb stem and '+' denotes the morpheme boundary. The practical use of these special characters is that 'A' "triggers" the application of certain morphographic rules (like the \(<s>\)-deletion presented above), and that '+' serves as a placeholder for certain characters that may be inserted in the surface form at the place of the morpheme boundary.

A collection of two-level rules can be translated into a collection of regular grammars, i.e., finite state machines, which in this case are finite state transducers (FSTs). Each of these FSTs accepts a certain language over the given alphabet of character pairs. All rules and hence all FSTs operate in parallel, which means that the language accepted by the collection of FSTs is the intersection of all individual languages. In order for a specific surface-lexicon mapping of a word to apply, the corresponding sequence of lexicon-surface character pairs must belong to that language.

In the conversion from two-level rules to FSTs a termination character pair is introduced, which is automatically set at the end of all surface strings treated by two-level rules. Therefore, all morpheme lexicon entries occurring at the end of a word (i.e., inflection endings) must include this character, or the character must be consumed separately by a grammar rule of the word grammar. The termination character can be defined by an SVOX system parameter (see [Tra93c]). In the current implementation it is set to the '+' character.

The SVOX system accepts two-level rules only in the form of FSTs. The following is an example of a two-level rule definition which contains the FST corresponding to the \(<s>\)-deletion rule as presented above. The rule-to-FST conversion was done using the software described in [Hub94].

\[ \text{:ALPHABET} \]
\[ 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' \]
\[ 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' \]
\[ 'y' 'z' '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' \]
3.3. Formalism of the Morpho-Syntactic Analysis

The two-level rule definition starts with the declaration of the alphabet, which is followed by zero or more finite-state automaton definitions. The alphabet and each automaton is ended by ‘*’.

The alphabet consists of lexicon-surface pairs, or, in the case of identical lexicon and surface characters, of this character only. Characters are denoted by a preceding apostrophe (‘), and the symbol ‘@’ stands for the empty character. The alphabet must always include the termination character pair (‘#’ in the given example). If the alphabet is omitted, a default alphabet is assumed which consists of character pairs for upper and lower case letters and special characters of the standard ASCII character set. A pair for the termination character is always included.

Each automaton declaration includes a description string, input class definitions, and a state transition table. In order to keep the automata as small as possible, character pairs that lead to the same state transitions are grouped into one input class. The default class comprises all character pairs not included in any other class. All classes must be given a unique number (usually in ascending order), which corresponds to the column number of the state transition table. Each row of the state transition table indicates one starting state and, for each input class, the corresponding next state. The state 0 denotes the failure state of the automaton: whenever any of the FSTs operating in parallel reaches the failure state, the given sequence of input character pairs is not allowed. State 1 is the starting state of the automaton and is also the accepting state. A sequence of input pairs is allowed if and only if all FSTs are in state 1 after the termination character pair has been consumed.

Figure 3.4 shows the mapping of the surface form <sitzt> onto the lexicon entries <sitzA-> and <s#> and the state transitions induced by the character pair sequence in the automaton corresponding to the given <s>-deletion rule. Section 3.4.1 will present the algorithm currently used in the SVOX system for looking up morphemes in a lexicon while simultaneously applying the two-level rule FSTs.

3.3.8 System Parameters

A number of system parameters control both the currently active SVOX knowledge bases and the processing of the morpho-syntactic analysis. All system parameters are declared in a separate file, which is loaded before any grammars and lexicons. Examples of system parameter declarations are

```
:PARAM SentGramFile  "cgram.dat"
:PARAM WordTermChar  "a"
:PARAM WordParseStrategy 0
:PARAM AddEdgePenalty 1000
:PARAM CheckLexValues :TRUE
:PARAM DrawS :FALSE
```

Each parameter declaration contains the parameter identifier and the initial value of the parameter, which may be of type string, integer,
or boolean. Parameter values may be changed interactively in the SVOX system. The full set of system parameters currently used is given in [Tra95c].

### 3.3.9 Morpho-Syntactic Knowledge Bases

Several data collections or linguistic ‘knowledge bases’ are needed by the SVOX system in order to perform the morpho-syntactic analysis. There are

- 3 grammars: a word grammar, which defines possible morphological structures of words, a sentence grammar, which defines possible sentence structures, and a lexicon editor grammar (see Section 3.5).
- 3 lexicons: a full-form lexicon, which contains words of closed word categories (so-called function words, like prepositions, articles, etc.), a morpheme lexicon, which contains morphemes and allomorphs (stems, inflection endings, prefixes, suffixes, etc.), and a submorphemic lexicon, which contains grapheme and phoneme clusters used for grapheme-to-phoneme conversion (see Chapter 4).
- 3 lexicon descriptions, one for each lexicon, which contain comments (for documentation purposes) and constituent declarations only and which are loaded before the corresponding lexicons, so that the values of lexicon entry features can be checked.
- 2 sets of two-level rules: a set of morphographic rules, by which graphemic surface forms are mapped onto lexicon entries, and a set of morphophonemic rules, by which phonemic lexicon entries are converted into the surface form of phonetic transcriptions.

All knowledge bases are usually stored in separate files (with names given by system parameters, cf. [Tra95c]), but it is also possible to

---

4Since these rules map the phonemic lexicon representations onto phonetic surface forms or vice versa, they could be termed both morphophonemic or morphographic rules. In this report, the term morphophonemic rules will be used, considering the fact that the primary purpose of these rules is to describe regular phonemic changes that occur at morpheme boundaries.

---

collect several knowledge bases (except the lexicon editor grammar and the lexicon descriptions) in data collections, which are especially useful for small experimental grammars andlexicons. An example of a data collection is

```plaintext
:GRAMMAR SENTGRAM
s  ==> np(?C,?N,?G) vp(?N) * 10
np(?C,?N,?G)  ==> persp(?C,?N,?G) * 1
np(?C,?N,?G)  ==> opt_art(?C,?N,?G)
               opt_art(?C,?N,?G)
               noun(?C,?N,?G) * 1
opt_art(?C,?N,?G)  ==> art(?C,?N,?G) * 1
opt_art(?C,?N,?G)  ==> * :INV
opt_adj(?C,?N,?G)  ==> adj(?C,?N,?G) * 1
opt_adj(?C,?N,?G)  ==> * :INV
vp(?N)  ==> verb(?N) *
:END

:GRAMMAR WORDGRAM
w  ==> noun(?,?,?,?,?) *
w  ==> verb(?,?,?,?,?)
noun(?C,?N,?G)  ==> nstem(?IC,?G)
noun(?C,?N,?G)  ==> nstem(?IC,?G)
verb(?N)  ==> vstem(?IC,?G)
:END

:LEXICON FULLEX
art(nom,sg3,neu)  "das"  "das"
art(acc,sg3,neu)  "das"  "das"  10
persp(nom,sg2,ni1)  "du"  "du"
:END

:LEXICON MORPHLEX
vstem(incl1)  "spielA"  "5p11;lA"
vstem(incl1)  "siztA"  "sitztA"
ve(incl1,sg2)  "st#"  "st#"
ve(incl1,sg3)  "t#"  "t#"
nstem(incl13,neu)  "kind"  "kind"
nstem(incl13,nom,sg3)  "#"  "#"
nstem(incl13,acc,sg3)  "#"  "#"
:END
```
would leave the full-form lexicon empty, but would still cause all other knowledge bases to be loaded from the separate files.

Connection of Word and Sentence Grammar

Due to the separation of word and sentence analysis and due to the separation of lexicons and grammars, a new view of grammar start and terminal symbols becomes necessary.

The start symbol of the sentence grammar (‘s’ in the sample data collection) is the main sentence constituent, which also becomes the root node of syntax trees. Terminal symbols of the sentence grammar are word categories (e.g., ‘noun’ and ‘art’). The definition of the term “word category” will be given below.

From the point of view of parsing, the terminal symbols of the word grammar are actually preterminal symbols i.e., all morpheme categories that occur in the morpheme lexicon. For technical reasons, there must be exactly one start symbol of the word grammar (‘w’ in the sample data collection). This start symbol simply produces all morphologically analyzable word categories. The sentence analysis is, however, not initialized with constituents of type ‘w’, but rather with constituents of word-category type.

The word category therefore plays an important role in the SVOX system. The set of word categories consists of all categories used in the full-form lexicon and of all categories that can be produced as direct visible subconstituents of the word grammar start symbol and which occur at least once in the head of a word grammar rule. The word grammar start symbol itself is not a word category symbol.

3.4 Morpho-Syntactic Processing

This section outlines the most important algorithms of the morpho-syntactic analysis, the lexicon access, the morphernization of words, and the chart parsing, and it presents some other details of the overall processing which are relevant for writing SVOX knowledge bases.
3.4.1 Lexicon Access

The two-level rules presented in Section 3.3.7 require a character-wise lexicon access for an efficient realization of the surface-to-lexicon mapping. This can be implemented most efficiently by using a character tree to store the lexicon.

Character trees allow a character-wise lookup of entries with an upper bound of complexity of \( O(n) \), where \( n \) = “length of entry”, i.e., the upper bound is independent of the number of entries in the lexicon. Character trees in the SVOX system are realized as shown in Figure 3.5, which depicts a character tree containing the verb stems “sitza+”, “sinkA+”, and “spielA+”, and the verb inflection endings “e#” and “st#”. Each node in the tree represents one character which belongs to one or more lexicon entries, and for each complete entry there exists a link from the last character node of the entry to a data package containing the entry type, morpho-syntactic features, phonetic transcription, etc. The “downward” pointer in the binary tree is the link to the first of all possible next characters, and the pointer “to the right” is the link to the next alternative character in the same position.

In order to look up an entry, the list of alternatives must be searched for the character at the current entry position (this takes at most \( m \) steps, with \( m = \) “size of alphabet”), and if the character has been found, the character at the next position of the entry must be searched on the next tree level. (In fact, a character tree is nothing but a finite-state machine which accepts the language that consists of all lexicon entries.)

The following two algorithms, ‘LookupChar’ and ‘LookupNullChar’, which are given in semi-verbal Modula-2 code, describe the character-wise lookup. ‘LookupNullChar’ looks up the empty (or null) character, which is needed in the morphemization algorithm of Section 3.4.2. Both algorithms return the new position in the tree and the lexicon entry (or, actually, a list of entries) found by the lookup, if the lookup was successful. Otherwise, NIL is returned for the new tree position and for the entry list.

```
PROCEDURE LookupChar (tree: CharTree; ch: CHAR;
VAR entries: LexEntry)
  (*newSubTree*) CharTree;

BEGIN
```

Figure 3.5: A character tree which stores the lexicon entries “sitza+”, “sinkA+”, “spielA+”, “e#”, and “st#”. 
t := tree^.firstSuccessor;
WHILE (t <> NIL) AND (t^.nodeChar <> ch) DO
  t := t^.alternative
END;
IF t <> NIL THEN entries := t^.lexEntryList
ELSE entries := NIL
END;
RETURN t
END LookupChar;

PROCEDURE LookupNullChar (tree:CharTree;
  VAR entries:LexEntry)
  : (*newSubTree*) CharTree;
BEGIN
  IF tree <> NIL THEN entries := tree^.lexEntryList
  ELSE entries := NIL
  END;
RETURN tree
END LookupNullChar;

Using the above procedures, an entire word or morpheme could be looked up as follows:

tree := rootNode;
i := 0;
WHILE (tree <> NIL) AND (i < Length(word)) DO
  tree := LookupChar(tree,word[i],entries);
  INC(i)
END;
IF (tree <> NIL) AND (entries <> NIL) THEN
  "entries found"
ELSE
  "no entries found"
END;

3.4.2 Morphemization of Words

This section presents the algorithm that is applied to look up all morphemes occurring in a given surface word form by consulting the morpheme lexicon and at the same time processing the two-level FSTs.

The basic idea of the algorithm, which is given below as procedure “FindMorphemes” in semi-verbal Modula-2 code, is to find morphemes starting from every position in the surface word form. This is achieved by hypothesizing lexical characters from surface characters (using the given character pair alphabet), and by testing whether the hypothesized sequence of lexical characters can be found in the lexicon and whether the corresponding pair sequence is allowed by all two-level FSTs.

Since words can only be composed of morpheme sequences without gaps, the only useful starting positions for morpheme lookups are the position 0 and all positions where a previously found morpheme ends. Moreover, the lookup must start at position 0 with the FST state tuple \((1, \ldots, 1)\), or at position \(n\) with an FST state tuple reached at position \(n\) at the end of a previously found morpheme. An FST state tuple in this context is one combination of the states of all FSTs operating in parallel. In order to implement these conditions, each position of the surface word form is associated with a set of possible FST state tuples, which are the starting state tuples for morphemes starting at this position. At the beginning of the morphemization, the only non-empty state tuple set is the set at position 0, which consists of the single state tuple \((1, \ldots, 1)\).

PROCEDURE FindMorphemes (word:ARRAY OF CHAR);
BEGIN
  (* it is assumed here that 'word' already contains the termination character at its end *)
  "initialize FST state tuple set to \{(1, \ldots, 1)\} in position 0, to \{\} in all other positions";
  FOR pos := 0 TO Length(word)-1 DO
    FOR "all FST state tuples currently in set of position pos" D0
      FindMorphemes1(word,pos,stateTuple,lexiconTree)
    END
  END
END
END FindMorphemes;

PROCEDURE FindMorphemes1 (word:ARRAY OF CHAR;
  surfacePos:INTEGER;
  startTuple:FSTTuple;
  treePos:CharTree);
BEGIN
  (* try to insert pair with empty surface character *)
FOR "all pairs with any lexical character lexCh and empty surface character" DO
newTuple := FSTTransition(startTuple, "lexCh:0");
newTreePos := LookupChar(treePos, lexCh, entries);
IF "not FST failure" AND (newTreePos <> NIL) THEN
  IF entries <> NIL THEN
    "insert found entries in word parsing chart";
    "include newTuple in set at position surfacePos"
  END;
  FindMorphemes1(word, surfacePos, newTuple, newTreePos);
END
END;
(* now try to look up lexical characters corresponding to current surface character *)
IF "not at word end" THEN
  surfCh := word[surfacePos1];
FOR "all pairs with given surface character surfCh and any lexical character lexCh" DO
  newTuple := FSTTransition(startTuple, "lexCh:surfCh");
  IF "lexCh = empty character" THEN
    newTreePos := LookupNullChar(treePos, entries)
  ELSE
    newTreePos := LookupChar(treePos, lexCh, entries);
  END;
  IF "not FST failure" AND (newTreePos <> NIL) THEN
    IF entries <> NIL THEN
      "insert found entries in word parsing chart";
      "include newTuple in set at position surfacePos+1"
    END;
    FindMorphemes1(word, surfacePos+1, newTuple, newTreePos)
  END
END
END FindMorphemes1;

3.4.3 Chart Parsing

Chart parsing [Kap73, Kay82, Ear72, Rus90] is a means for the efficient parsing of sentences by context-free and context-sensitive grammars. The core of all chart parsers is a data structure, the chart, in which intermediate parsing results (found constituents, so-called passive edges) and hypotheses about constituents to find (so-called active edges) are stored. Besides the chart, an agenda stores remaining tasks to be done (e.g., hypotheses to be verified).

Different parsing strategies can be realized using a chart parsing framework. In the SVOX system, an Earley-type top-down parser [Ear72] was realized by Sascha Brawer [Bra94]. While this top-down parsing is more efficient than other parsing strategies, it is useful only if a full parse can be found. Since word analysis in the SVOX system requires that a word be completely parsable, the top-down strategy is usually applied for this purpose. However, in sentence analysis, the input sentence often proves to be unparseable, sometimes because a sentence is not correct in the sense of German syntax, and sometimes because the sentence grammar does not handle certain possible German sentence structures. Therefore, a different parsing strategy is applied for sentence parsing.

This sentence parser, implemented by the author, is a bottom-up parser which simply accumulates in the chart all constituents (passive edges) that can be found anywhere in the sentence. It uses a very simple and general algorithm, and unlike other bottom-up chart parsers, it does not store any active edges (constituent hypotheses) in the chart and it has no predefined direction of operation (left-to-right or right-to-left).

In order to apply the bottom-up algorithm, the grammar must be binarized, i.e., there must not be more than two subconstituents in the right hand side of any grammar rule. To achieve this, grammars are binarized upon loading by creating new auxiliary (invisible) grammar rules where necessary. For example, the grammar rule

\[
np(\text{C}, \text{N}, \text{G}) \rightarrow \text{opt\_art}(\text{C}, \text{N}, \text{G}) \\
\text{opt\_adj}(\text{C}, \text{N}, \text{G}) \\
noun(\text{C}, \text{N}, \text{G}) \ast 1
\]

which is a sentence grammar rule from the sample grammar given in Section 3.3.3, is automatically split into the two grammar rules

\[
np(\text{C}, \text{N}, \text{G}) \rightarrow \text{np\#1}(\text{C}, \text{N}, \text{G}) \\
noun(\text{C}, \text{N}, \text{G}) \ast 1
\]

\[
\text{np\#1}(\text{C}, \text{N}, \text{G}) \rightarrow \text{opt\_art}(\text{C}, \text{N}, \text{G}) \\
\text{opt\_adj}(\text{C}, \text{N}, \text{G}) \ast 0 : \text{INV}
\]
The bottom-up parsing algorithm as applied for the sentence analysis works as follows: Initially, all word constituents occurring in the sentence are stored in the chart. Additionally, for all empty grammar rules and for empty words (if any), empty edges are initialized in all vertices (shown only schematically in Figure 3.6). The agenda in the parsing algorithm keeps track of all initial and newly created edges, and for all entries in the agenda the algorithm simply tries to combine the corresponding edge with neighboring edges to the left and right, according to the given binary grammar rules. If a combination is successful, a new edge for the head constituent is inserted in the chart, if no identical edge already exists. New edges are also noted in the agenda. The precondition for the combination of two edges is that they occur as right hand side in a grammar rule and that their feature terms can be unified with the corresponding terms of the grammar rule. (If the algorithm is applied to the word analysis, which can be done in the SVOX system, the compatibility of the FST states of two neighboring edges must also be tested before combining them.)

The bottom-up parsing algorithm is given below in semi-verbal Modula-2 code. It assumes that grammar rules are given as pointers to data structures (r-) defining the head constituent type, the head features, a first subconstituent type with first subfeatures and a second subconstituent type with second subfeatures. The edges are pointers to data structures (e-, eR-, eL-) containing the constituent type, the corresponding features, and the starting and ending vertex in the chart.

```
PROCEDURE BottomUpParse;
  (* assumes that the chart is initialized with word or morpheme category edges and with all empty grammar rules and that all initial edges are noted in the agenda *)

PROCEDURE NewEdge (newCons:Cons; start,end:INTEGER;
  ruleFeat:Features; uvars:Bindings);
BEGIN
  newFeat := "create new features from ruleFeat using the variable bindings in uvars";
  "search edge from start to end with newCons/newFeat";
  IF "no identical edge found" THEN
    "insert edge newCons/newFeat from start to end in the chart and note it in the agenda"
```

Figure 3.6 shows the chart after parsing the sentence “das Kind spielt” using the grammar and lexicon of Sections 3.3.3 and 3.3.4. The chart is a lattice structure with n + 1 vertices where n = “length of input”, and with edges leading from start vertices to vertices with higher or equal number. Vertices are set between adjacent words and at the beginning and end of the sentence. An edge of type ‘cons’ from vertex a to vertex b indicates that the words between a and b can be interpreted as a constituent of type ‘cons’.
BEGIN
   e := "next edge removed from agenda";
   WHILE e <> NIL DO
      (* treat rules with a single subconstituent *)
      FOR "all rules r with r^\.subcons1 = e^\.cons
         and with r^\.subcons2 = NIL" DO
         IF "unification possible between
e^\.feat and r^\.subfeat
         by generating variable bindings uvars" THEN
            NewEdge(r^\.headcons,e^\.start,e^\.end,
                        r^\.headfeat,uvars)
         END;
      END;
      (* combine e with edges to the right of e *)
      FOR "all rules r with r^\.subcons1 = e^\.cons and
         with r^\.subcons2 <> NIL" DO
         FOR "all edges eR with eR^\.start = e^\.end
            and with eR^\.cons = r^\.subcons2" DO
            IF "unification possible between
e^\.feat and r^\.subfeat1 and between
eR^\.feat and r^\.subfeat2
            by generating variable bindings uvars" THEN
               NewEdge(r^\.headcons,e^\.start,eR^\.end,
                           r^\.headfeat,uvars)
            END;
         END;
      END;
      (* combine e with edges to the left of e *)
      FOR "all rules r with r^\.subcons2 = e^\.cons" DO
         FOR "all rules eL with eL^\.end = e^\.start
            and with eL^\.cons = r^\.subcons1" DO
            IF "unification possible between
eL^\.feat and r^\.subfeat1 and between
e^\.feat and r^\.subfeat2
            by generating variable bindings uvars" THEN
               NewEdge(r^\.headcons,eL^\.start,e^\.end,
                           r^\.headfeat,uvars)
            END;
      END;
   END;

THE TREATMENT OF UNPARSABLE SENTENCES

If a sentence cannot be parsed (i.e., if no edge for the start symbol of the sentence grammar exists in the chart after parsing), an artificial parse tree is created by finding the way through the chart with minimal total penalty (using the penalty values of individual edges, which are computed from their subedges and from the grammar rule penalties). This can easily be done by applying a dynamic programming technique. All constituents of this minimal path are then simply interpreted as direct descendants of an artificially generated sentence constituent. Thus, even if a sentence cannot be parsed completely, the later TTS steps will still be provided with useful syntactic grouping information, and the sentence analysis therefore exhibits a "graceful degradation" behavior.

A very important detail in the finding of the shortest path through the chart is the fact that an additional edge penalty is added to all edges, which is given in the system parameter 'AddEdgeQuality' and which should usually be chosen to be greater than the maximum penalty occurring in a grammar rule. Without this additional penalty, the path through A and B would have less penalty than the path through X for an applied grammar rule X → AB with a positive penalty value, which is not desired. The additional edge penalty leads to the preference of higher constituents over the combination of lower constituents.

3.4.4 Interplay of Sentence and Word Analysis

This section presents some important details of the morpho-syntactic processing of sentences.

Text Input Conversions: Some character-wise input conversions are done before the morpho-syntactic analysis. These conversions are
defined by system parameters (see [Tra95c]) and comprise simple character substitutions and the conversion of all characters to lowercase characters. (If desired, the conversion to lowercase characters can be omitted by changing a system parameter.)

The set of characters accepted by the word analysis (after the afore-mentioned conversions) is defined completely by the alphabet of the morphographemic two-level rules. All characters that appear as surface characters anywhere in the alphabet are accepted, all other characters are ignored.

Sentence end: In a linguistic sense, a sentence is finished at the end of a sequence of words that denotes a syntactically and semantically closed statement. In TTS systems, more practical definitions must be found to define the sentence end. In the SVOX system, a sentence terminator character (‘!’, ‘?’, or ‘;’, as given in the system parameter ‘TerminatorChars’) marks the end of a sentence if the input token (sequence of characters not interrupted by blanks) which contains the terminator character cannot be analyzed as a whole by the word analysis. Thus, if the full-form lexicon contains the entry “etc.”, the ‘;’ in the token “etc.” is not regarded as sentence end.

Input tokens are treated from right to left in order to find sentence terminators. For each terminator character found in this right-to-left process, a blank character is hypothesized first after, then before the terminator character until the token portion from the left end to the blank can be analyzed as a whole or until all terminator characters have been treated. The token portion from the left end to the last hypothesized blank will then be regarded as one word to analyze. The remainder of the token will be considered a sentence terminator if it consists of one terminator character only, otherwise the right-to-left treatment will be repeated for the remaining token.

For example, if the full-form lexicon contains the word “x?y?z” and the TTS input contains the token “x?y?z.x?y?z”, the ‘;’ after the first ‘;’ is treated as a sentence end which separates the two words “x?y?z” and “x?y?z”.

The full stop is sometimes treated in a wrong way. For example, “12,” will always be interpreted as an ordinal number within a sentence. By some plausible modifications, this interpretation can be changed if necessary: “12th,” will be treated as the cardinal number “12” which is located at the sentence end, and “12th” or “12th,” will be treated as an ordinal number located at the sentence end.

Word analysis: Words are first segmented into morphemes according to the procedure given in Section 3.4.2. The word parsing chart is initialized with the found morphemes (and actually also with full-form entries found for the current word). The word is then parsed according to the word grammar. All acceptable solutions of the word analysis are used to initialize the sentence chart. Acceptable word analyses are all different readings (i.e., all different word category/feature combinations that span the entire word) which do not have a penalty value greater than $p_{\text{min}} + \text{WordQualityRange}$, where $p_{\text{min}}$ is the minimal penalty of all solutions and ‘WordQualityRange’ is given as a system parameter. For the acceptable readings, a morphological tree is built, which is passed on to the sentence analysis.

Sentence analysis: Based on the results of the word analysis, the sentence analysis parses the sentence and produces a morpho-syntactic parse tree. This tree describes the sentence structure as well as the morphological structure of the words. (Section 3.4.3 outlines the procedure for building a syntax tree in the case of an unparseable sentence.)

Generation of phonetic surface form: After the sentence analysis, the phonetic surface form (i.e., the phonetic transcription) of each word in the morpho-syntactic tree is generated from the phonemic lexicon representation of the morphemes or full-forms that span the word by applying the morphophonemic two-level rules.

Conversion of the morpho-syntactic tree: In connection with the generation of the phonetic surface forms of the words in a sentence, the morpho-syntactic tree is converted into a syntax tree, which does no longer contain the morphological structure of the words, but only more shows the syntactic sentence structure. The terminal elements in this tree are entire words in their graphemic and phonetic surface representation. The syntax tree is the basis for the subsequent TTS steps (accentuation and prosodic phrasing).
3.5 Lexicon Editor

Although the lexicons in the SVOX system are text files, which can be modified with any text editor program, a specialized lexicon editor has been added to the system in order to simplify lexicon maintenance. The lexicon editor was implemented by Boris Sigrist and Minh Trang Nguyen [SN93].

Figure 3.7 displays the basic appearance of the editor. The user can define new entries and delete or modify old entries. In order to enter a new word, the user must type in the graphemic and phonetic representation of a word or morpheme, then choose the word or morpheme category (out of a given set), and, based on this choice, select appropriate values for all required morpho-syntactic features out of the sets of allowed values. The allowed values are simply taken from constituent declarations, which are mandatory for all categories used in the lexicon editor.

On demand, the phonetic transcription of the entry is proposed by the editor using the grapheme-to-phoneme conversion described in Chapter 4, and the user can edit the transcription and also invoke the acoustic synthesis of the transcription.

The most critical feature values to select are the inflection classes. In order to support the choice of these classes, all relevant forms of the inflected word are automatically displayed for the current selection of the inflection class value, as shown in Figure 3.7.

The word and morpheme categories accessible by the editor and the displaying of inflected forms is completely defined in a lexicon editor grammar. This grammar, like all other grammars, contains grammar rules and, optionally, constituent declarations. An example of such a grammar (the grammar used to produce Figure 3.7) is

```
! Adjective stem
! ------------
EDCONS(AS,ADJ_STEM) => *
```

All rules in this grammar have a head constituent ‘EDCONS’, whose first feature value defines a category to edit (i.e., its constituent identifier) and whose second feature provides a colloquial description of the constituent in the form of an atomic term.

Rules with empty right hand side define constituents which may be edited but for which no inflection display should be produced. A sequence of rules for the same category and for the same description which immediately follow each other define a category with inflection display, and the right hand sides of the rules define the generation of the actual inflected forms. All right hand sides consist of a word stem and an inflection ending.

Currently, only this form of non-empty rule is allowed. Moreover, all features in the stem must be variables, and exactly one feature of the inflection ending must be a variable which also occurs in the stem. This linking variable indicates the feature that defines the inflection class.

---

5Unfortunately, the editor is currently available in the VAX/VMS version of SVOX only.
Figure 3.7. Layout of the lexicon editor. The graphemic form of the new noun stem, boot+, has been entered, the word category and morphological features have been selected from the sets of allowed values. An inflection display helps to select the correct values of the singular and plural class. The phonetic form has been proposed by the system.

<table>
<thead>
<tr>
<th>category</th>
<th>AS</th>
<th>P2S</th>
<th>NS</th>
<th>NOUN_STEM</th>
<th>lex:</th>
<th>FLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sing. cl.</td>
<td>SK2</td>
<td>SK3</td>
<td>SK4</td>
<td>SC: boot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plur. cl.</td>
<td>PK5</td>
<td>PK6</td>
<td>PK7</td>
<td>bootes,-s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gender</td>
<td>N</td>
<td></td>
<td>Neutr</td>
<td>boot,-e</td>
<td>boot</td>
<td></td>
</tr>
<tr>
<td>Surf.cat</td>
<td>?</td>
<td>N</td>
<td>Noun</td>
<td>booten</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC:</td>
<td>boote</td>
<td></td>
<td></td>
<td>boote</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(entry inserted)

The stem constituent for the generation of inflected forms is given as the stem currently being entered into the editor, and the inflection endings are taken from the morpheme lexicon. If several solutions of the inflection form generation exist, all forms are displayed as in the example of the singular class and the other four forms display the case variants of the plural class.

In the example, the first four inflected forms represent all case variants of the singular class, and the other four forms display the case variants of the plural class.
Chapter 4

Grammar-Based Grapheme-to-Phoneme Mapping

4.1 Introduction

As has been shown in Chapter 3, the mapping of a graphemic representation of words and sentences onto a phonemic representation in the SVOX system is mainly based on the access to a full-form and to a morpheme lexicon, in which graphemic items are stored together with the corresponding phonemic representation. A set of grammars (word and sentence grammar, finite-state transducers for two-level rules) sort out all allowed grapheme-to-phoneme mappings for words and select the best of all allowed mappings.¹

Chapter 3 described the mapping of graphemic sentences onto the phonetic transcription for “ordinary” words and sentences. There are, however, some special problems arising with grapheme-to-phoneme

¹It is very common in the TTS literature to speak of grapheme-to-phoneme conversion. However, since this conversion is fully bidirectional in the SVOX system, the term grapheme-to-phoneme mapping will be preferred in this report.
mapping, especially the pronunciation of abbreviations, numbers (digit sequences), and novel words, i.e., words that are not stored in a full-form lexicon or which cannot be analyzed morphologically. In conventional TTS systems the first two problems are usually treated in a special text preprocessing module, in which abbreviations and digit sequences are expanded to full orthographic or phonetic forms. However, in many cases in German and other inflected languages, the proper pronunciation of (especially cardinal) numbers and abbreviations depends on the syntactic context in which they occur. The appropriate place for grapheme-to-phoneme mapping of such items therefore appears to be not before, but within or together with the syntactic analysis. This is a position also taken in [LLG93].

The following sections will briefly outline the SVOX grapheme-to-phoneme mapping approach for the aforementioned problems and they will show that no new formalisms or procedures apart from those of Chapter 3 are needed to handle (at least the most common of) them. The treatment of abbreviations, numbers, and even novel words could therefore be realized as “by-products” of the very general and flexible morpho-syntactic analysis. Moreover, all these mappings exhibit the same bidirectionality of grapheme-to-phoneme/phoneme-to-grapheme mapping as the regular morpho-syntactic analysis.

No general text preprocessing has therefore been integrated in the SVOX system. For the prototype application of reading aloud weather forecasts, the text preprocessing consists only of an application-specific text formatting, which, for example, inserts sentence ends and modifies or deletes certain text passages.

### 4.2 Abbreviations

German abbreviations such as “Uno” [ˈʊ:no], “Lkw” [ˈleːkɐː-ˈveː], or “WHO” [ˈhjuː-ˈɛː] are usually spoken as a single word if they obey the phonotactic rules of German (i.e., if they are pronounceable as a word), otherwise they are spelled. (There are, of course, exceptions to this rule, e.g., “USA” [ˈjuː-ˈɛː-lɐi]). The pronunciation of such abbreviations can easily be stored in the full-form lexicon of the SVOX system. If, however, they are not lexically stored, these abbreviations are submitted to the grapheme-to-phoneme mapping procedure described in Section 4.4. In the SVOX system, this mapping is based on the phonotactic structure of German words, such that an ordinary word pronunciation is assigned to pronounceable abbreviations. In addition to this word pronunciation, a default rule always generates the spelled pronunciation. However, since this default rule bears a high penalty value, the spelled pronunciation form is only chosen if the abbreviation is not pronounceable as a regular word.

More complex German abbreviations, like “z. B.” (“zum Beispiel” / “for example”), “z. Z.” (“zur Zeit” / “for the time being”) or “u. v. a.” (“und viele andere” / “and many others”) can be handled by full-form lexicon entries for the individual particles of the abbreviations and by a few general sentence grammar rules that bind the appropriate particles together. The following data collection shows how the above abbreviations can be treated in the SVOX system.

```
:LEXICON FULLLEX
  abbria(u) "u." "<2nt"
  abbr2a(zB) "z." "t_s2m"
  abbr2b(zB) "b." "bt_15p;1.0"
  abbr2a(zE) "z." "t_seu;4"
  abbr2b(zE) "z." "t_sea;1t"
  abbr3a(uva) "u." "<2nt"
  abbr3b(uva) "v." "f1;16"
  abbr3c(uva) "a." "<and6r6"
:END

:GRAMMAR SENTGRAM
  abbr --> abbria(?) *  [one particle]
  abbr --> abbr2a(?Id) abbr2b(?Id) *  [two particles]
  abbr --> abbr3a(?Id) abbr3b(?Id) abbr3c(?Id) *  [three particles]
:END
```

Each abbreviation particle carries an identifier which denotes the entire abbreviation. The sentence grammar rule simply interprets all particles belonging to the same abbreviation as a single constituent and thereby selects the correct pronunciation in ambiguous cases. In this approach, one general grammar rule is required for all different numbers of particles in abbreviations.
As a side-effect of the special treatment of full stops, the purpose of which is to find sentence ends and which has been described in Section 3.4.1, abbreviations are treated identically whether the individual particles are separated by blanks or not. For example, “z. B.” and “z. B.” are both automatically regarded as consisting of two particles (unless, of course, there exists a separate full-form lexicon entry for “z. B.”).

Instead of using only one constituent type for abbreviations (‘abbr’ in the example above), it is possible to use more precise specifications for different types of abbreviations (e.g., abbreviations for prepositional groups, for adverbial groups, for noun phrases, etc.). This can be useful for distinguishing homographic abbreviations, as in the case of “Meier u. a. fanden: …”, which is expanded to “Meier und andere fanden unter anderem folgendes: …” (“Meier and others found among other things the following: …”).

A data collection that could handle these cases could be designed as shown below. In conjunction with an appropriate sentence grammar, the abbreviations in the previously given sentence could be distinguished. The example also shows how different pronunciations can be generated depending on the case of the constituent that contains the abbreviation.

```
:LEXICON FULLLEX
abbr2a(untand,adv) "u." "<2nt4"
abbr2b(untand,adv) "a." "'<and6rf6a"
abbr2a(ua,(cnp,?)) "u." "<2nt"
abbr2b(ua,(cnp,com)) "a." "'<and6rf6"
abbr2b(ua,(cnp,gen)) "a." "'<and6r6e"n"
abbr2b(ua,(cnp,dat)) "a." "'<and6r6n"
abbr2b(ua,(cnp,acc)) "a." "'<and6r6n"

:END

:GRAMMAR SENTGRAM
abbr(?Cons) ==> abbr2a(?Id,?Cons) abbr2b(?Id,?Cons) *
ad ==> abbr(adv) *
cnp(?Case) ==> abbr((cnp,?Case)) *

:END
```

Abbreviations in this example are interpreted as being of adverbial function (‘ad’) or as being the second part of a coordinated noun phrase (‘cnp’). The first feature in the abbreviation particles is the abbreviation identifier, and the second feature is a term denoting the syntactic function in the sentence. If the abbreviation is interpreted as a coordinated noun phrase (‘cnp’), a structured term must be used for denoting both the type and the case of the constituent as a single feature.

### 4.3 Numbers

Digit sequences which represent cardinal or ordinal numbers or fractions are converted into a phonemic representation in SVOX by treating the digit sequence as a “word”, the individual digits as “morphemes” and by applying a set of word grammar rules that define the pronunciation structure for different types of numbers. Thus, the morpheme lexicon includes entries for all digits and also for some digit combinations (especially for the German numbers between 13 and 99, in which the pronunciation order is opposite to the digit order) in all possible pronunciation variants, and a special number grammar, which is a part of the word grammar, defines the appropriate sequence of the different digit/pronunciation elements.

This process of finding the pronunciation of digit sequences is somewhat simplified in the current SVOX solution by a special “placeholder morpheme” (‘*’), which a two-level rule optionally inserts on the lexical level after each digit of the surface level. The purpose of this placeholder is to assume the pronunciation of different intervening elements, such as “tausend”, “thousand” or “hundert”/“hundred”, where needed in the pronunciation of the entire number.

The following data collection shows the principle of converting the cardinal numbers 1 to 9999 into their pronunciation form (except for the cases in which ‘0’ digits occur in non-final positions or where ‘1’ occurs as first digit, for which some additional structure rules would be needed).

```
:LEXICON MORPHLEX
term "1" "1"

dig(s) "3" "'dra_1"
dig(m) "3" "'dra_1"
```
4.3. Numbers

All lexicon entries are given in a weakly and a strongly accented form (features ‘w’ and ‘s’). The grammar rules select an appropriate pronunciation variant and the accent pattern for the whole number in parallel. For the number 3335, the corresponding morphological tree generated by the SVOX system is shown in Figure 4.1.

The number grammar used in the SVOX system is too large to be presented here. It currently treats cardinal and ordinal numbers and fractions with an integer magnitude in the range from 0 to 9999999999. For ordinal numbers, which are inflected like German adjectives, all possible readings are passed on to the sentence grammar, which selects the appropriate form.

The pronunciation of German year numbers in the range 1100 up to 1999 differs from that of ordinary cardinal numbers (they are pronounced in hundreds only, e.g., “fünfzehnhundertachtundsechzig” for “1578”). Since these numbers occur in other syntactic positions than regular cardinal numbers (they do not occur in adjective position), the correct pronunciation could be selected by the sentence grammar (not yet realized in SVOX). The additional word grammar rules

```
dig(w) "35" "'fynf<2nt'"dra_1sic

ddig(s) "35" "'fynf<2nt'"dra_1sic

numpart(w.hundert) "#" "'h2nd4t"
numpart(s.hundert) "#" "'h2nd4t"
numpart(w.tausend) "#" "'ta_uz6nt"
numpart(s.tausend) "#" "'ta_uz6nt"
```

```
:END

:GRAMMAR WORDGRAM
w => num * [make 'num' a word category]
um => num1 term *
um1 => num1 term #
umnt => ddig(s) * [1..9]
um1 => numnt #
umnt => ddig(s) * [11..99]
um1 => numnt #
umth => dig(w) numpart(w.hundert) numt * [211..999]
um1 => numth #
umth => dig(w) numpart(w.tausend) numh * [2211..9999]
```

```
:END

Figure 4.1: The pronunciation of the digit sequence “3335” as found in the SVOX system by a structural analysis of the sequence as in the regular morphological analysis. Placeholder “morphemes” (∗) are optionally inserted after each digit by a two-level rule. The number grammar makes use of these pronunciation elements where necessary. The parse tree shows the constituent identifiers only, feature terms are not displayed.
```
with ‘ddig1’ denoting two-digit numbers starting with 1, show how year numbers could be pronounced as desired.

*Prices and times* are in many languages written in a form not directly convertible into pronunciation. For instance, “Fr. 13.50” must be pronounced as “13 Franken 50”. This could be attained in SVOX by the following rules, in which the period assumes the pronunciation of the currency, and the pronunciation of the currency abbreviation is empty.

```plaintext
:w => ynum *
ynum => ynuml term *
ynuml => ddig1(w) numpart(w,hundert) ddig(m) *
```

Different strategies are applied to convert a grapheme sequence into a corresponding phoneme sequence, for example:

- The classical approach is the establishment of *letter-to-sound rules* in the form of an ordered set of context-sensitive rewrite rules. These rules attempt to convert graphemic segments into corresponding phonemic segments depending on the graphemic and phonemic context of the segments. Usually, a minimal morphological analysis is performed on a word before applying letter-to-sound rules, during which prefixes and suffixes are stripped off in order to isolate word stems. An example of a letter-to-sound rule is a rule applied in the MITTalk system [AHK87, p. 60]: “When a precedes r, and r is not followed by either a vowel or another r within the same morph. a is pronounced AA (e. g., far, cartoon) unless preceded by the segment WW (e. g., warble, warp, war, wharf quarter).”

- Letter-to-sound rules in the form of ordered sets of rewrite rules can be learned automatically using *symbolic learning* methods, as described in e. g. [Coi91]. The learning approach described therein derives a set of rewrite rules capable of correctly pronouncing a given phonetic dictionary. By learning the pronunciation of a set of words with minimal context specifications, the basic pronunciation rules of a language are learned, which are expected to also generate an appropriate pronunciation for new words.

### 4.4 Unknown Words

#### 4.4.1 Introduction

The SVOX system follows a strongly lexicon-based approach to the conversion of written orthographic words into phonetic form, and there is clearly a general trend in TTS systems toward the utilization of large lexicons. Nevertheless, as has been pointed out in [CC190], there will always be a remainder of words (especially proper nouns) which are not known to the TTS system (often termed *novel words*). For these cases, and as long as the available TTS lexicons are small, a rule-based grapheme-to-phoneme mapping is a necessity. Moreover, such rule-based mappings can be of great help in establishing pronunciation dictionaries.

A minor drawback of the above solutions to the pronunciation of numbers is that the entire pronunciation is realized as one phonetic word. In order to change this, special phonetic symbols would have to be introduced which would have to be changed into word separators in later TTS steps.
• A similar approach, but using neural networks (i.e., a subsymbolic learning scheme) instead of symbolic learning procedures, was first realized in the famous NetTalk system [SR6, SR87].

• Several methods exist for the pronunciation of novel words by re-using parts of grapheme/phoneme mappings given in phonetic dictionaries, i.e., in analogy to other words. For example, in the rhyme analogy method presented in [CC190], the name “Plotsky” is pronounced in analogy to the name “Trotsky”, i.e., the identical right hand part of the word (“otsky”) is reused for the pronunciation of the new word, and for the initial part of the new word, ordinary letter-to-sound rules are applied.

In the analogy procedure in [DN91], a pronunciation lattice is set up for the novel word in which all possible pronunciation variants are stored for substrings of the novel word which can also be found as parts of words in a pronunciation dictionary. The optimal path through this lattice yields the pronunciation of the entire novel word. The optimal path is determined by the path length (i.e., number of different dictionary entries used) and by the summed frequency count of all pronunciation parts (i.e., the frequency count of the used dictionary entries in the language).

4.4.2 Pronunciation of Novel Words in SVOX

For the design of grapheme-to-phoneme conversion for novel words in the SVOX system, the major question was how to integrate such a conversion in the framework of the morpho-syntactic analysis. Since inflection endings, prefixes, and suffixes are supposed to be completely stored in the morpheme lexicon, the only missing items for which a direct grapheme-to-phoneme mapping must be applied are word stems. One solution therefore offered itself very naturally: the extension of the analysis of sentence and word structures to the analysis of word stems by means of a stem grammar and a lexicon of grapheme and phoneme clusters, which together define the graphotactic and phonotactic structure of German word stems [Tra93]. A first implementation of this concept was carried out by Susanne Werner [Wer93].

The following data collection shows the basic scheme of grapheme-to-phoneme conversion for unknown stems:

:GRAMMAR SENTGRAM
  s ➞ np(TC,TN,TG) vp(TN) * 1
  np(TC,TN,TG) ➞ persp(TC,TN,TG) * 1
  np(TC,TN,TG) ➞ noun(TC,TN,TG) * 1
  vp(TN) ➞ verb(TN) np(TC,TN,TG) * 1
:END

:GRAMMAR WORDGRAM
  w ➞ noun(?,?,?,?) * 1
  w ➞ verb(?) * 1
  noun(TC,TN,TG) ➞ cnst(TIC,TG) ne(TIC,TN,TG) * 1 [noun]
  cnst(TIC,TG) ➞ enst(?,?) cnst(TIC,TG) * 1
  cnst(TIC,TG) ➞ enst(TIC,TG) * 1
  enst(TIC,TG) ➞ rep_pref() nst(TIC,TG) * 1
  enst(TIC,TG) ➞ rep_pref() nst(?,?,?) nsuff(TIC,TG) * 1
  rep_pref() ➞ * 1 : INV
  rep_pref() ➞ pref() * 1
  nst(TIC,TG) ➞ nstem(TIC,TG) * 1 [lexical stem]
  nst(TIC,TG) ➞ gstem * 1
  nst(TIC,TG) ➞ fstem * 1
  verb(TN) ➞ evst(TIC) ve(TIC,TN) * 1 [verb]
  evst(TIC) ➞ pref() vst(TIC) * 1
  evst(TIC) ➞ vst(TIC) * 1
  vst(TIC) ➞ vstem(TIC) * 1 [lexical stem]
  vst(TIC) ➞ gstem * 1
  vst(TIC) ➞ fstem * 1

[stem grammar]
[R1] gstem ➞ gicc lvow gfc1 su * 1
[R2] gstem ➞ gicc lvow gfc2 su * 1
[R3] gstem ➞ gicc cvow su * 1
[R4] gstem ➞ gicc cvow gfc1 su * 1
[R5] gstem ➞ gicc spevc su * 1
[R6] fstem ➞ usyl13 aclisy1 * 1
[R7] fstem ➞ usyl13 asyl usyl * 1
usyl13 ➞ * 1 : INV
usyl13 ➞ usyl * 1
usyl13 ➞ usyl usyl * 1
usyl3  ==>  usyl usyl usyl * 1
usyl  ==>  usyl usyl * 1
usyl  ==>  usyl usyl * 1
usyl  ==>  usyl usyl * 1
uosyl  ==>  ficc scv * 1  [unacc. open syll.]
uosyl  ==>  ficc scv * 1  [unacc. open syll.]
ayanl  ==>  aoyl aoyl * 1
ayanl  ==>  aoyl aoyl * 1
ayanl  ==>  aoyl aoyl * 1
ayanl  ==>  aoyl aoyl * 1
acslayl  ==>  ficc lcv * 1  [acc. open syll.]
acslayl  ==>  ficc lcv * 1  [acc. closed syll.]

:END

:LEXICON FULLLEX
persp(nom,sg3,mas)  "er"  "<e;""":END

:LEXICON MORPHEX
pref  "be"  "b6"
ve(infc11,sg3)  "t#"  "t#"
ne(infc12,acc,sg3)  "#"  "#"

:END

:LEXICON SubMORPHLEX
gicc  "$"  "<'"  100  [glottal stop]
gicc  "t"  "t"
gicc  "k1"  "k1"
gicc  "w"  "w"
gicc  "st"  "1t"n
lvow  "a"  "a;
svow  "e"  "e;
svow  "o"  "o;
cvow  "au"  "a_u"
cvow  "ae"  "e;
gfci  "$"  "1"
gfccc  "$"  "c"
gfccc  "$"  "i"
spevc  "och"  "qz"
"s"  "s"
su  "$"  "$"
"s"  "$"

:END

ficc  "m"  "m"
ficc  "n"  "n"
ficc  "t"  "t"

ficc  "$"  "m"

ficc  "$"  "m"

:END

:TWOLEVEL MOGRAPH

:ALPHABET
'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l'
m 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x'
'y' 'z' 0  '1' '2' '3' '4' '5' '6' '7' '8' '9'
'.' ',' '/' '%' '-' '#' '+' '/' '0' '1' '2' '3' '4' '5' '6' '7' '8' '9'

:END

:TWOLEVEL MOPHON

... corresponding to MOGRAPH ...

:END

The actual grapheme-to-phoneme conversion of stems is done by entries in a submorphic lexicon, which stores the graphemic and phonemic representation of vowel and of consonant clusters used in the stem grammar. The stem grammar itself is merely a part of the regular word grammar. The submorphic lexicon, however, is a separate lexicon (i.e., not a part of the morpheme lexicon) for reasons of clarity and efficiency: For all words in the morpho-syntactic analysis, a regular morphological analysis is attempted first by using the morpheme lexicon only. If this analysis fails, the analysis is simply repeated with a parallel lookup of lexicon entries in both the morpheme and the submorphic lexicon in the morphemization procedure described in Section 3.4.2. The additional submorphic items found in this second lookup then enable the stem analysis and grapheme-to-phoneme mapping. Since the stem analysis requires much more processing time than the regular morphological analysis (on the order of 1 second per unknown word on a VAXstation 4000/60), it is only invoked when actually needed. From a logical point of view, the separation of the two analyses and of morpheme and submorphic lexicon would not be necessary.
In the word grammar given above, verb stems can be prefixed ('evst'), and noun stems can be extended by prefixes and suffixes ('enst') and by composition ('enst'). Furthermore, noun and verb stems can either be lexically stored ('nstem' and 'vstem'), or they can be analyzed as (unknown) generic German stems ('gstem') or foreign stems ('fstem').

Genuine German stems are monosyllabic, consisting of an initial consonant cluster ('gicc'), a vowel or diphthong in the nucleus, and a final consonant cluster ('gcfc1' and 'gcfc'). The stem syllable may be followed by one of the special suffixes ('su') 'e', 'el', 'en', or 'er'. This basic structure is described by the rules R1…R5. The initial consonant cluster can phonemically consist of a glottal stop only, with no correspondence in the graphemic representation. This choice is, however, strongly discouraged by the high penalty value in the corresponding lexicon entry. (Since the empty word is not usable in the morpheme lexicon, a placeholder is applied (8), which corresponds to the null character on the graphemic and phonemic surface level. A corresponding character pair '8:0' must therefore be included in the alphabets of the morphographic and morphophonemic rules.)

The rules R1 and R2 distinguish between final consonant clusters that require a long closed vowel ('lbow'; as in 'flur'/'flu:x', 'tag'/'ta:k', or 'log'/'lo:k]) and consonant clusters that require a short open vowel ('svow'; as in 'stel'/'stil', 'schul'/'ʃa:l', or 'bank'/'ba:k]). R3 and R4 describe the structure of stems like 'traum'/'tra:m', 'zieh'/'tsiː', 'scc'/'scː', or 'kal'/'kaːl] with a complex nucleus (diphthongs or vowels that are graphemically denoted as long, e.g., 'au', 'au', 'ih'). R5 treats the special cases like 'dach'/'daːx', 'fuchs'/'foːks', or 'dachh'/'da:xh] with a final consonant cluster starting with 'ch' after one of the vowels 'a', 'o', or 'u'. (In other cases, the 'ch' is pronounced as [k], which is considered the regular case in the given grammar.)

Foreign stems (mostly of Greek and Latin origin) are usually polysyllabic and contain much smaller consonant clusters than genuine German stems. If the location of the accented syllable in a foreign stem is known, the grapheme-to-phoneme mapping can be done quite reliably. Otherwise, the mapping must guess this location, which is one of the major problems. In the above example, only two forms of foreign stems are set up in R6 and R7: the accent location is supposed to lie on the last syllable if this syllable is closed. otherwise the accent is put on the penultimate syllable. Before the accented syllable, up to three unstressed open or closed syllables may occur.

R6 and R7 correctly treat words like "kanal" / [ˈkaːnəl], "zitrone" / [ˌzi.trɔnə], and "kurve" / [ˈkʊʁvə]. Additional rules would be needed to treat, for instance, cases with double consonants ("akkurat" / [əˈkuːʁat]) or words with special suffixes like "ik" or "or", which change the stress position of the stem, depending on the presence or absence of additional suffixes or inflection endings.

Figure 4.2 shows the result of the analysis of the sentence "er bestellt Mineralwasser" ("he orders mineral water") by means of the given grapheme-to-phoneme rules. All stems in the sentence are supposed to be unknown.

The grapheme-to-phoneme conversion in the described manner is quite similar to the approach taken in [HMSZ93], and it is also related to the grapheme-to-phoneme approaches by analogy. The stem grammar currently applied in the SVOX system is too large to be presented here, but it obeys the same overall structure as the example rules presented above. Some problems are still unsolved, and for some cases known solutions have not been implemented yet. The main problem of the grammar-based approach is the fine tuning of penalty values for the stem grammar, the penalty values are highly important for the selection of a plausible pronunciation. One of the main problems in this context is that the German language allows compound words, for which it is often difficult to find an appropriate segmentation into morphemes automatically, if one or several stems are unknown: Polysyllabic foreign stems can often be analyzed as a series of German stems as well, which leads to a rather different pronunciation. The selection of the appropriate solution requires a highly sophisticated system of stem grammar rules and penalty values.

The following properties characterize the SVOX grapheme-to-phoneme mapping for novel words:

- The grapheme-to-phoneme conversion is a natural extension of the regular morpho-syntactic analysis, and it could be implemented without adding any new formalisms or procedures. It is perfectly possible to only let parts of words be converted by rules (e.g., only
one of a series of stems in a compound noun).

- The grapheme-to-phoneme conversion can still benefit from the two-level rules (like, e.g., umlaut substitution or the treatment of the \(<e>\)-deletion in verb stems like “handel!”).

- Unlike in classical letter-to-sound rewrite rules it is easily possible to access very wide structural contexts in the grapheme-to-ph
ome conversion, if, for instance, a suffix changes the pronunciation of the beginning of a word (as in “photo”/[ˈfɔːto] and “photographie”/[fɔtɔˈfi])

- It is possible to let the sentence syntax decide on the pronunciation of a novel word in ambiguous cases, e.g., “sucht”/[ˈzuːxt] (a verb with stem “such”) and “sücht”/[ˈzuːt] (a noun with stem “sücht”), since all different readings together with their possibly different pronunciation are passed on to the sentence analysis.

- Due to the fully declarative nature of the entire grapheme-to-phoneme mapping it is possible to reverse the grapheme-to-

  phoneme conversion into a phoneme-to-grapheme conversion by simply exchanging the graphemic and the phonemic representa-

  tions in the lexicons and by exchanging the morphographic and the morphophonemic rules.

- Morphological decomposition of words, stress assignment, stem origin and stem boundary detection based on the analysis of spec-

  ific consonant clusters, and grapheme-to-phoneme conversion are all done in parallel. It is therefore not necessary to decide on the order of the different conversion steps, which proved to be a problem in classical rewrite rules as stated in [Kla87, p. 772] (order of stress assignment and phoneme prediction, left-to-right or right-

  to-left processing). However, some of these problems are converted into problems of the assignment of suitable penalty values in the

  SVOX stem grammar.

4.5 Summary

From Chapter 3 and from the previous sections in this chapter it becomes clear that the entire grapheme-to-phoneme mapping, be it for
regular words, for abbreviations, for numbers, or for novel words, is performed in the same uniform scheme: a lexicon provides all possible pronunciation variants of the parts of which a respective item (a “word”) is composed, and then a grammar which reflects the internal structure of the item selects possible sequences of these pronunciation forms. Where necessary, allowed sequences are further ordered according to a numeric quality criterion, and the best solution is taken as the final result.

Not all of the possibilities shown in this chapter have been realized in the SVOX system so far, but all given examples work in the described manner. The aim of this chapter has mainly been to show possible solutions to well-known problems which fit smoothly into the overall framework of the morpho-syntactic analysis of the SVOX system.

Chapter 5

Accentuation and Prosodic Phrasing

5.1 Introduction

Accentuation and prosodic phrasing, i.e., the relative degrees of prominence of syllables in an utterance and the subdivision of an utterance into speech groups, largely depend on the semantics of an utterance, as has already been pointed out in Chapter 2. The semantic content of an utterance is only partly reflected in its lexical and syntactic structure. However, the approach taken in the SVOX system closely follows approaches to the derivation of accentuation and phrasing from the syntactic structure of an utterance, which are expressed for example in [CH68, Sei84] and especially for German in [Kip66, Bie66]. The algorithms for accentuation and phrasing currently applied in the SVOX system, which will be described in this chapter, are actually adapted versions of the algorithms presented in [Kip66, Bie66].

Accentuation and phrasing currently clearly belong to the weakest points in the SVOX system (if counting the number of “errors”), and a redesign of the algorithms in close connection with a redesign of the sentence grammar will become necessary in future. Not too much emphasis will therefore be laid on the present solutions.
5.2 Syllabification

A very important preparatory step for all subsequent prosody processing modules is the syllabification of the phonetic transcription. i.e. the assignment of a syllable boundary between each pair of neighboring syllable nuclei (vowels or diphthongs).

Some morphologically motivated syllable boundaries are already set by the morpho-syntactic analysis, but the remaining boundaries must be assigned according to phonetic criteria, as given in, e.g., [Dud74].

The following rules are applied in the SVOX syllabification (in the given order):

1. If a syllable boundary is already set between two nuclei, this boundary is the resulting boundary.
2. A boundary is set directly between two nuclei if no consonants are present.
3. A boundary is set between two identical consonants.
4. A boundary is set before the special consonants or consonant clusters [pf], [pf], [tsv], [ts], [fpr], [ftr], [hl], [br], [dr], [gl], [gr], [bl], [kn], [kr], [kv], [pf], [pl], [pr], [fl], [jl], [fr], [fm], [jn], [fp], [ft], [f], [ts], [tr], [l], [l], if the clusters immediately preceed a syllable nucleus. The consonant group before a nucleus is searched for the clusters in the given order. (Some of these clusters actually represent German stem-initial clusters, and the boundaries before these stems should rather be set by the morphological analysis already. In future, this might be corrected.)
5. A boundary is set before the last consonant in the consonant group between the nuclei.

For example, the above rules would syllabify “Verbesserungsvorschlag” [fɛɡˈ buzəɾʊŋɡsˌfɔkəfɪsk] as [fɛɡˈ buzərʊŋsˌfɔkəfɪsk] which is the correct syllabification in this case. Accent markers that occur in the phonetic transcription are ignored by the above rules, but the resulting syllable boundaries are set before any accent markers.

5.3 Accentuation

5.3.1 Introduction

Accentuation in a phonological sense distinguishes different meanings of different utterances with the same segmental content by means of assigning different levels of prominence to syllables. For example, the German words “umfahren” (“to run down”) and “umfahren” (“to drive around”) are distinguished by different stress patterns, which result in different syllable prominence patterns if the words are uttered.

It is very common in phonology to denote the primary accent of an utterance by ‘1’, the secondary accent by ‘2’, the tertiary by ‘3’ etc. (A different notation is used in, e.g., [Sel84]).

If a single word is uttered in isolation, the primary accent of this utterance is put on the main stress position of the word. For example, the afore-mentioned accent distinction is realized as 1 umfahren vs. 1 umfahren

When several words together form an utterance, their prominence, i.e., the prominence of the syllables carrying the main word stress, is mutually weighted depending on the semantics and the syntax of the utterance. For example, the sentence “ich bin gekommen” / “I have come”, if uttered as a simple piece of new information, receives the accentuation 1 ich bin gekommen

whereas in the more complex sentence “ich bin nach Hause gekommen” / “I have come home” the accentuation would be

1 2 ich bin nach Hause gekommen
The presence of the prepositional phrase “nach Hause” causes a shift of the main accent of the utterance from the word “gekommen” to the word “Hause”, that is, “Hause” in the second utterance is equally prominent to the word “gekommen” in the first utterance. The word “gekommen” in the second utterance is clearly less prominent than the same word in the first utterance, and it is also less prominent than “Hause”. These relative levels of prominence are expressed by the accent values.

Semantic differences in syntactically and lexically identical utterances may result in entirely different accentuation patterns. For example, the sentence “Peter fliegt nach Paris” (“Peter flies to Paris”) is accented as

\[
\begin{align*}
2 & \quad 3 & \quad 1 \\
\text{Peter fliegt nach Paris}
\end{align*}
\]

if it is uttered “out of the blue”, that is, as a completely new piece of information. If, however, the sentence should point out the fact that Peter does not drive to Paris but that he takes the plane, or that it is not John who flies to Paris but Peter, the corresponding accentuation Patterns would be

\[
\begin{align*}
3 & \quad 1 & \quad 2 \\
\text{Peter fliegt nach Paris}
\end{align*}
\]

\[
\begin{align*}
1 & \quad 3 & \quad 2 \\
\text{Peter fliegt nach Paris}
\end{align*}
\]

In such contrastive accentuation patterns, every word of an utterance can receive the primary accent in an appropriate semantic context. However, what is usually investigated is a “neutral” accentuation pattern, i.e., the pattern which requires the least specific semantic context and which can more or less reliably be derived from the syntactic structure of a sentence. This neutral accentuation on a sentence-by-sentence basis is normally applied in today’s TTS systems.

5.3.2 Accentuation Principles

The accentuation algorithm given in [Kip66], from which the algorithm in the SVOX system was derived, is based on the following assumptions:

- Some words are accetable (mostly so-called content words like nouns, verbs, adjectives, and adverbs). All accetable word categories initially receive a primary accent on their word stress position, which corresponds to the accentuation of these words if uttered in isolation.

- The actual accentuation procedure works in cyclic fashion, i.e., from the leaves of the morpho-syntactic tree to the sentence node. In each cycle, the subconstituents of one node are combined together, and the accentuation pattern within this new constituent is determined according to two rules:

  1. Each constituent category of the morpho-syntactic tree is defined to be either left-accented or right-accented, which means that either the leftmost or the rightmost of the 1-accents in the new constituent retains its primary accent, whereas other primary accents are reduced to 2, which in turn reduce all 2-accents to 3, and so on. This rule is known in the linguistic literature as the nuclear stress rule, due to which the nucleus of a constituent remains primary-accented, whereas all other accents are reduced.

  2. For each cycle, some accent patterns are changed according to a general rhythmic stress shift rule: The pattern 3 2 1 (with arbitrary intervening unstressed syllables) is changed to 2 3 1, and the pattern 1 2 3 is changed to 1 3 2.

After the application of the cyclic accentuation rules, the whole sentence contains one primary accent, the sentence accent.

Although Kiparsky’s work may seem very old already, the nuclear stress rule and the rhythm rule presented above are still a vital part of much newer accentuation theories, for instance [Sel84].

5.3.3 Accentuation Algorithm in SVOX

The accentuation scheme according to Kiparsky has been adapted by Ruth Rothenberger and the author to the special needs and the special syntax tree generated in the SVOX system [TR88]. The following is a description of this SVOX accentuation algorithm.
Word Accent Pattern

For each word in the syntax tree, a numeric accent pattern is extracted from its phonetic transcription. To this end, all accent marks (') are counted which occur to the left of each syllable nucleus (up to the next nucleus or the beginning of the word) and the number of accent marks directly denotes the accent strength.

If there is more than one primary accent (1) in the word, only the leftmost primary accent remains whereas other primary accents are changed to secondary accents (2). This corresponds to the usual left-accentuation in German words. In [Kip66], the accentuation of individual words is treated by cyclic accentuation rules which operate on the morphological structure of the words. Since the morphemes are no longer separated after the application of the morphophonemic two-level rules in the current morpho-syntactic analysis in SVOX, the cyclic accentuation rules cannot be applied to single words and the accentuation for most compound words is obtained by the afore-mentioned ad-hoc principle. For more complicated words such as numbers, the word accentuation pattern is generated by the word grammar already, as shown in Chapter 4.

According to the above rules: the word ["ba:n-baibern-te"] will produce the accentuation pattern 1 0 2 0.

Initial Sentence Accents

Some word categories (articles, prepositions, personal pronouns, coordination particles, and others) are declared to be unstressed. The accent pattern of these words is changed such that the primary accent becomes a very weak accent (99 in the current system), and secondary and tertiary accents become correspondingly weaker accents (100, 101, etc.). The reason for this treatment will become clear in Section 5.5.

All other word categories are acceptable and retain the accent value 1 on the primary word accent. All other accents within these words are increased by 2, such that the strongest non-primary accent gets the value 4. For example, the pattern 1 0 2 0 will become 1 0 4 0. The decreasing of minor word accents is done in order to prevent these accents from interfering with the overall sentence accentuation.

Cyclic Accentuation Rules

The cyclic accentuation rules work on a modified syntax tree, in which the leaves, i.e., the graphemic/phonemic representation of words, are initially replaced by the corresponding word accentuation patterns. All leaves of this accentuation tree are accent values of one syllable of the utterance.

For the sentence “das kleine Kind sieht die Blumenbeete” ["das 'klai-nə 'kmat 'zi:t 'di; 'blu-mən-'be-te"] (“the little child sees the flower beds”), a possible initial accentuation tree is shown in Figure 5.1.

The accent values in this tree are modified by traversing the tree from the leaves to the root and by applying a set of accentuation patterns for the constituents encountered in this traversal. These accentuation patterns all specify a possible subtree of the syntax tree (i.e., a possible constellation from the current constituent toward the leaves) and an action to be carried out if the pattern matches the given syntax tree.
For each constituent type, a collection of accentuation patterns may be specified, which are compared with the syntax tree in the order of declaration. And if a pattern matches the syntax tree, the corresponding action is carried out, and no other patterns are applied any more for the same constituent. If a pattern matches the tree in several places, only the first (from left to right) place is considered. If no pattern for a specific constituent type can be applied, some patterns are used which hold for any constituent type. If still no match is found, nothing is changed in the accentuation tree.

Accentuation patterns have the structure of ordinary accentuation trees, but additionally, four wildcard symbols may be used and actions may be specified together with the tree nodes. Although desirable, no feature specifications are currently possible in the nodes of the accentuation patterns.

The pattern wildcard symbols are:

• matches any (possibly empty) sequence of nodes on the same tree level.

**: like ‘•’, but is stronger if it occurs together with ‘•’, in that it matches as many nodes as possible.

+: matches exactly one arbitrary node of the tree.

++: matches a (possibly empty) sequence of nodes on a top-down path in the tree.

Accentuation patterns must always fully match all subconstituents of specified nodes, so that wildcard symbols must be used frequently to match all subconstituents.

In the present accentuation algorithm, there is only one action that can be specified:

acc1, m, acc2: matches an accent value acc1 and marks it (‘m’) with the new value ‘acc2’.

The meaning of this action is that it usually marks the nucleus of a constituent for the subsequent application of the nuclear stress rule.

5.3. Accentuation

If the accent pattern permits (indicated by a flag associated with the whole pattern), the stress reduction principle of the nuclear stress rule is applied after the matching of a pattern (i.e., once for each constituent or cycle). The stress reduction has been implemented as in [Kip66]: If a marked accent is set to the new value ‘acc2’, all other accents of level ‘acc2’ within the same constituent are numerically increased by 1, then so also are all accents of level ‘acc2 + 1’, ‘acc2 + 2’, etc., until a gap in this sequence of accent levels is encountered. For example, if the stress reduction principle is applied to the accents (2, 1, 4, 1, 99) starting with the marked new value 1 (not included in the list), the resulting accents will be (3, 2, 4, 2, 99).

Examples of accentuation patterns are shown in Figure 5.2. The first two patterns realize the simple right- and left-accentuation of the constituents ‘vp’ and ‘vc’ according to [Kip66]. The third pattern states that within a noun group (‘ng’) with a noun (‘n’) at the end, the primary stress remains on the noun and all other accents (e.g., on adjectives preceding the noun) will be reduced by the subsequent application of the reduction principle. The fourth pattern states that within a sentence (‘s’), the rightmost noun group (‘ng’) below some undefined subconstituent (+) at the right end of the sentence will receive the sentence
to this rule, some accent constellations are changed to new ones. These changes are

\[
\begin{align*}
3 & 2 & 1 
\rightarrow 2 & 3 & 1 \\
2 & 2 & 1 
\rightarrow 2 & 3 & 1 \\
1 & 2 & 3 
\rightarrow 1 & 3 & 2 \\
1 & 2 & 2 
\rightarrow 1 & 3 & 2
\end{align*}
\]

In these patterns, weaker accents (value 0 or greater than 3) may intervene, which remain unchanged. In [Kip66], the patterns containing two accents of level 2 are not treated because they do not occur if the nuclear stress rule is applied to binary syntax trees only, as is the case in that work. In the current SVOX system, however, rather flat trees can be generated which lead to accent patterns with more than one secondary accent, for which the rhythmic stress shift (or rather rhythmic stress reduction in that case) should be applied as well.

The stress shift rule is applied for all constituents except for some constituent types given in a special list. Currently, all main and subordinate clause constituents are excluded from the application of the stress shift rules.

### Postcyclic Accentuation Rules

Some accent rules are executed after the traversal of the syntax tree. These postcyclic rules are again given as accentuation patterns, which are always applied to the root node of the sentence. Unlike the cyclic patterns, the postcyclic patterns are applied in all positions where a match can be found. As in the case of the cyclic patterns, a flag indicates whether the stress reduction principle should be applied.

In the SVOX system, postcyclic patterns are currently used to lower the accent strength on finite verbs. These are usually too strongly accented by the application of the nuclear stress rule, because in the syntactic structure generated in SVOX the finite verb of a sentence is a direct descendend of the sentence node. This often results in a secondary accent on the finite verb. The postcyclic accentuation rules change accent values of 2 and 3 on all finite verbs below main clause constituents.
in the sentence to a value of 4. Other accents remain as before, that is, the accent reduction principle is not applied in the postcyclic accentuation rules.

Example of SVOX Accentuation

For the sentence “ein grosses atlantisches Sturmtief verlagert sich heute nach Osten” (“a large Atlantic cyclone is moving towards the east today”), the current SVOX system generates the syntax tree shown in Figure 5.4. The accentuation of the sentence is derived in the steps shown in Figure 5.5.

5.4 Prosodic Phrasing

5.4.1 Introduction

Prosodic phrasing is the division of an utterance into several speech groups, or phrases. The boundaries between these groups are phonetically realized by pauses, by special melodic patterns, and/or by lengthening of the syllable(s) before the boundary. The prosodic phrases correspond, to a certain degree, to syntactic phrases, but they are not always identical with syntactic phrases.

It is often easier to say which phrasings are incorrect (i.e., ungrammatical) than to say which phrasings are allowed and appropriate in specific contexts. For example, for the sentence “my brother has given me a nice birthday present”, phrasings such as

*(my) (brother has) (given me a) (nice birthday) (present)*

are easily recognized as ungrammatical (indicated by **). To provide correct phrasings is more difficult. For the sentence “Jane gave the book to Mary” Selkirk [Sel84, p. 293] gives the following correct phrasings:

1 Selkirk actually only discusses intonational phrasing, i.e., the division of an utterance into melodic groups. In the SVOX system, however, no distinction is currently made between melodic and rhythmic phrases.
5.4. Prosodic Phrasing

1: (Jane gave the book to Mary)
2: (Jane) (gave the book to Mary)
3: (Jane gave the book) (to Mary)
4: (Jane gave) (the book) (to Mary)
5: (Jane) (gave the book) (to Mary)
6: (Jane) (gave the book) (to Mary)

The number of phrases into which an utterance is divided depends on the fluency and rapidity of the speaking style. Example 6 given above may represent a slow dictation style, whereas Example 1 occurs in a fluent conversation.

5.4.2 Phrasing Principles

The phrasing algorithm given by Bierwisch in [Bie66] is used in the SVOX system with only slight modifications. The algorithm uses some information on the accentuation of the sentence and is based on the following principles:

- Initially, all leaves of the syntax tree are temporary phrases.
- Temporary phrases are combined with neighboring temporary phrases into longer phrases in cyclic fashion, i.e., from the leaves of the syntax tree to the root, according to the following criteria:
  - Clitic elements, i.e., completely unaccented words (function words) are enclosed in that neighboring phrase to which they are syntactically more closely bound.
  - Temporary phrases which do not contain a minimum number of accented words or a minimum number of syllables may not remain independent and are combined with that neighboring phrase to which they are syntactically more closely bound. The minimum number of syllables and accents necessary for a phrase to be independent is determined by a tempo parameter: A larger parameter value leads to longer phrases.
– After the main accent of a sentence, no phrase boundaries occur.

The algorithm formulated in [Bie66] does not directly operate on the syntax tree, but in the following way:

- An initial phrase boundary is set between each pair of adjacent words of a sentence. The strength of this boundary corresponds to the level of the first common ancestor node of the two words in the syntax tree. Syntactically closely connected words are therefore separated by an initial boundary, loosely connected words by a strong boundary.

- Phrase boundaries are then deleted cyclically according to certain criteria, i.e., by going from weaker boundary levels to stronger boundary levels and within the same level, from left to right, and deleting a boundary if certain conditions based on accent and syllable counts are satisfied.

- A postcyclic rule deletes all phrase boundaries after the main accent of the sentence.

The boundaries that remain undeleted by this process are then defined to be the actual phrase boundaries of the sentence.

5.4.3 Phrasing Algorithm in SVOX

In the SVOX algorithm, the phrasing algorithm was implemented as described in the following paragraphs. Phrase boundaries are denoted by \( \#n \) in this section. \( n \) indicates the separation strength of a boundary, with 0 being the strongest separation and 1, 2, etc. being successively weaker separations. (In the phonological representation described in Section 5.6, phrase boundaries will be denoted by \( \#\{n\} \).)

Initial Phrase Boundaries

Initial phrase boundaries are set around the full sentence and between each pair of adjacent words according to the rules given below. Where several rules apply, only the strongest boundary is maintained (i.e., the boundary with the least numeric index). Generally, \#0 and \#1 are undeletable boundaries and will produce a pause in the phonetic interpretation of the phrase boundaries. The boundaries with index \( >2 \) are deletable, but \#2 is only produced by special initializations (see below). Based solely on the syntactic structure, \#3 is the strongest sentence-internal initial boundary.

1. \#0 is set before and after the whole sentence.

2. A boundary \#n is set between each pair of words, where n is the level of the closest common ancestor node of the two words in the syntax tree. The level of the root of the syntax tree is defined to be 3, and the level number is increased by 1 with each new tree level toward the leaves.

3. Special boundary indexes are set around some constituent types. These special cases are given by patterns which describe the constituent type and the left and right boundary index to be used. Additionally, a direct predecessor constituent type is specified which must be present in order for the special indexes to apply.

4. For each punctuation character in the orthographic input sentence, a boundary \#1 is set.

An example of a special phrase boundary index pattern (initialization rule 3) used in the SVOX system is the pattern ‘(v, ks, 2, 100)’, which means that around a finite verb (‘v’) which is located directly below a main clause node (‘ks’) the boundaries 2 (left) and 100 (right) are set. The value 100 is simply used to denote the absence of any special initialization. The effect of this rule is to set a stronger boundary before the finite verb than after it. This initialization, which conforms to observations of natural speech, would automatically be set by initialization rule 2 if a classical grammar rule of the form ‘s → np vp’ were applied in the SVOX sentence grammar. Since this is currently not the case, a special initialization pattern is used to initialize the stronger boundary to the left of the finite verb.

Similar patterns are currently used to set the boundary before coordinations (“und”/“and”, “oder”/“or” and others) to the index 2.
Cyclic Deletion of Phrase Boundaries

Some of the initial phrase boundaries are deleted in a cyclic fashion, i.e., from higher indexes to lower ones. Unlike the original algorithm in [Bie66], the SVOX algorithm uses two complete cyclic applications of boundary deletions. A first one to combine unaccented (clitic) words with one of their neighboring temporary phrases, and a second one, in which temporary phrases are combined with larger phrases based on rhythmic criteria (accent and syllable count and a speaking rate parameter).

In each of the cyclic procedures, indexes are treated from the highest occurring index down to index 2, and for each index, all boundaries of this index are treated from left to right. For each boundary, a decision is made whether it can be deleted or not. A boundary is deleted by converting it into an ordinary word boundary.

In order to evaluate the decision criteria, the following characterizing features are computed for each boundary:

\[
\begin{align*}
\text{ind}_l \text{ and ind}_r : & \text{next boundary index to the left and right of the current boundary} \\
\text{nsyl}_l \text{ and nsyl}_r : & \text{number of syllables between the current boundary and the next boundary to the left and right} \\
\text{nacc}_l \text{ and nacc}_r : & \text{number of accented words between the current boundary and the next boundary to the left and right; accented words are those which have been judged acceptable by the accentuation rules and which carry an accent value smaller than the special weak accent level of unaccentable words (i.e., < 99)}
\end{align*}
\]

In the first cyclic phrase boundary deletion (melting of clitic words), a boundary \#n is deleted if at least one of the following conditions is satisfied:

- \( \text{nacc}_l = 0 \) and \( n \geq \text{ind}_l \)
- \( \text{nacc}_r = 0 \) and \( n > \text{ind}_r \)

In the second cyclic phrase boundary deletion (rhythmic melting of temporary phrases), a boundary \#n is deleted if at least one of the following conditions is satisfied:

- \( (n > \text{lowind} \text{ or } p \geq 2) \) and \( n \geq \text{ind}_l \) and \( n \geq \text{ind}_r \) and \( \text{nacc}_l \leq q \)
  where
  \[
  q = \begin{cases} 
  p + 1 & \text{if } \text{nsyl}_l \leq 2 \\
  p & \text{if } 2 < \text{nsyl}_l < 5 \\
  p - 1 & \text{if } \text{nsyl}_l \geq 5 
  \end{cases}
  \]
- \( (n > \text{lowind} \text{ or } p \geq 2) \) and \( n \geq \text{ind}_l \) and \( n > \text{ind}_r \) and \( \text{nacc}_r \leq q \)
  where
  \[
  q = \begin{cases} 
  p + 1 & \text{if } \text{nsyl}_r \leq 2 \\
  p & \text{if } 2 < \text{nsyl}_r < 5 \\
  p - 1 & \text{if } \text{nsyl}_r \geq 5 
  \end{cases}
  \]

In both cases, ‘lowind’ is the lowest deletable boundary index (2), and \( p \) is a parameter defining the desired degree of phrasing. Usually, \( p = 1 \) or \( p = 2 \) is appropriate. Higher values of \( p \) delete more, lower values delete fewer boundaries.

Postcyclic Deletion of Phrase Boundaries

The postcyclic phrase boundary deletion has been implemented in SVOX in the following way: within each domain delimited to the left and right by \#0 or \#1 (and not containing \#0 or \#1 itself), all phrase boundaries after the main accent are deleted. The main accent in this context is defined as the rightmost of all accents of the strongest level (i.e., lowest numeric value) within the domain.

The phrase boundaries that remain after this postcyclic deletion procedure are the actual phrase boundaries of the sentence.

Example of SVOX Phrasing

Figure 5.6 shows the syntax tree which is generated by the current SVOX system for the sentence “ein Tief, das über dem nahen Atlantik liegt, verlagert seinen Schwerpunkt morgen nach Frankreich” (“a low which lies above the near Atlantic will move its center to France tomorrow”). Figure 5.7 depicts the application of the phrasing algorithm for this sentence. The accentuation shown in Figure 5.7 is the pattern
generated by SVOX, but the special weak accent level for “accentless” words (99) is replaced by 0.

5.5 Accent Normalization

The accentuation algorithm presented in Section 5.3 may lead to a very large number of different accent levels within longer sentences and within highly structured trees. In accordance with two principles given in [Bie66], a normalization of the accent values is done in SVOX after the phrase boundaries have been determined. These two principles are:

- Accentuation by accent reduction (nuclear stress rule) should be done separately within each subordinate and main clause of a sentence in order to limit the number of accent levels to a phonetically interpretable range.

- In [Bie66], a phonological representation of the intonation (the sentence melody) is derived from the accentuation and phrasing information by picking out the two main accent levels within each phrase and associating accents of these levels with major pitch movements. The last accent of the highest accent level within each phrase is associated with the “anchor point” of the intonation pattern of the phrase, which may be non-terminal (“progredient”), or terminal.

In the SVOX system, no phonological representation of the intonation is explicitly generated (the sentence melody is directly derived from the accentuation pattern). The above considerations are therefore realized as the following reordering of the accentuation levels:

- Within each phrase, the rightmost accent of the strongest level within the phrase is reset to a primary accent (level 1), i.e., it becomes the phrase accent. This is done even if the strongest level is the special weak level for “accentless” words (99).

- All other accents are strengthened as much as possible while maintaining their relative prominences (stronger/weaker/equal). This is, however, done only for the strongest accent within each word.

Figure 5.6: Syntax tree generated by SVOX for the sentence “ein Tief, das über dem nahen Atlantik liegt, verlagert seinen Schwerpunkt morgen nach Frankreich”. The tree is shown in indentation form down to the full-word level and without features. The levels in this tree structure, which are indicated at the top, are the basis of the phrasing algorithm.
Initial boundaries according to tree levels:
#0 <0> #6 <3> #5 <0> #6 <0> #8 <0> #10 <0> #10 <2> #6 <4>
#4 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #4 <0> #6 <1> #0

Special boundary before verb:
#0 <0> #6 <3> #5 <0> #6 <0> #8 <0> #10 <0> #10 <2> #6 <4>
#2 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Punctuation boundaries:
#0 <0> #6 <3> #1 <0> #6 <0> #8 <0> #10 <0> #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Cyc 1. after deletion level 10:
#0 <0> #6 <3> #1 <0> #6 <0> #8 <0> #10 #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Cyc 1. after deletion level 8:
#0 <0> #6 <3> #1 <0> #6 <0> #8 <0> #10 #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Cyc 1. after deletion level 6:
#0 <0> #3 #1 <0> #6 <0> #8 <0> #10 #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Cyc 2. after deletion level 10:
#0 <0> #3 #1 <0> #6 <0> #8 <0> #10 #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Cyc 2. after deletion level 8:
#0 <0> #3 #1 <0> #6 <0> #8 <0> #10 #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Cyc 2. after deletion level 6:
#0 <0> <3> #1 <0> #6 <0> #8 <0> #10 #10 <2> #6 <4>
#1 <0> 4 <0> #4 <0> #6 <2> #4 <2> #4 <0> #6 <1> #0

Result:
#0 ein Tief #1 das über dem nahen Atlantik liegt #1 verlagert seinen Schwerpunkt #4 morgen nach Frankreich #0

Figure 5.7: The SVOX phrasing algorithm applied to the sentence “ein Tief, das über dem nahen Atlantik liegt, verlagert seinen Schwerpunkt morgen nach Frankreich”. Initial phrase boundaries between words are set according to the syntactic connection strengths, and some of the boundaries are then deleted in two cyclic deletion procedures. Words are denoted by the word accent patterns in angular brackets, and phrase boundaries by ‘/\n’. Only those steps which modify the temporary phrasing are shown.

5.5. Accent Normalization and only for accented words (main accent level < 99). Within each word, the exact relative accent levels are maintained in this step.

- The accent levels are then further restricted to the levels 1…4 and 0, which are phonetically interpretable levels. All remaining weaker accents in the range from 5 up to the special level for unaccented words (99) are set to level 4, and all levels greater or equal to that special level are set to 0.

For the examples given in Sections 5.3 and 5.4 (“ein großes atlantisches Sturmtief verlagert sich heute nach Osten” and “ein Tief das über dem nahen Atlantik liegt, verlagert seinen Schwerpunkt morgen nach Frankreich”), the accentuation and phrasing generated by the SVOX algorithms are

#0 <99> <3 0> <0 4 0> <2 5> #2 <0 4> <99> <2 0> <99> <1 0> #0
#0 <99> <3> #1 <99> <99> <99> <99> <5 0> <0 2> <4> #1
<0 4 0> <99 0> <2 0> <4> #4 <2 0> <99> <1 0> #0

By the accent normalization procedure, the accentuation of these two examples is changed to

#0 <0> <2 0> <0 3 0> <1 4> #2 <0 3 0> <0> <2 0> <0> <1 0> #0
#0 <0> <1> #1 <0> <0> <0> <3 0> <0 1 0> <2> #1
<0 2 0> <0 0> <1 0> <4> #4 <2 0> <0> <1 0> #0

If an utterance only consists of a single unstressed word (like “er” or “ihnen”), the word stress position of this “unaccented” word, which is at the special weak level for unaccented words, is set to a primary accent by the above normalization procedure, as happens in natural utterances.
5.6 Phonological Representation

The results of accentuation, prosodic phrasing, and accent normalization, together with the phonetic transcription of the words of an utterance form the phonological representation of this utterance, which will be interpreted phonetically and acoustically by the phono-acoustical model described in Chapter 6. For the examples of Sections 5.3 and 5.4, the following phonological representations are obtained in the current SVOX system:

\[
\begin{align*}
\text{ ("ein grosses atlantisches Sturmief verlagert sich heute nach Osten")} \\
\#\{0\} & \text{ [P]} \text{ [ain [1]tlf.} \#\{1\} \text{ [P]} \text{ das [y-]bor dem [3]na-c-wan} \\
\text{ [at-]1lan-tik [2]likt} \#\{1\} \text{ [P]} \text{ fcr-[2]lan-gort za-tan} \\
\text{ ("ein Tief, das über dem nahen Atlantik liegt, verlagert seinen Schwerpunkt morgen nach Frankreich")}
\end{align*}
\]

In this representation, the following special symbols appear:

\( \#\{n\} \) stands for a phrase boundary, with ‘\#\{0\}’ denoting the sentence break before and after the sentence, ‘\#\{1\}’ denoting a sentence-internal pause, and ‘\#\{n\}’ with \( n > 1 \) denoting other phrase boundaries (without pause). If not explicitly set, ‘\#\{0\}’ is automatically assumed before and after the sentence.

(X) is a phrase type mark, which is set at the beginning of a phrase. ‘(P)’ denotes a phrase with non-terminal and ‘(T)’ a phrase with a terminal intonation pattern. Question phrase types have not yet been defined in SVOX. If a phrase type is omitted, it is set to ‘(T)’ in the last phrase of the utterance and to ‘(P)’ in all other phrases.

\( [n] \) stands for an accent of level \( n \) on the following syllable, with the intended interpretation of the accent levels being as follows:

\[ [1] \text{ denotes the main phrase accent, which is the anchor point of the phrase intonation pattern.} \]
\[ [2] \text{ denotes a pitch accent, i.e., an accent with a major pitch movement if it stands before the main phrase accent. Otherwise, it denotes a stressed syllable but without major pitch movement.} \]
\[ [3] \text{ denotes a non-pitch accent on the main stress syllable of a word.} \]
\[ [4] \text{ denotes the weakest accent, which usually occurs on secondary and tertiary word stress syllables.} \]
\[ [0] \text{ denotes completely accentless syllables. This value is set for all syllables without explicit accent mark.} \]

Although the phonological representation should be a minimal representation, there is some redundancy in the above accentuation scheme, as far as its phonetic interpretation is concerned. It could be mapped onto the more phonetically motivated accentuation scheme of Kohier [Koh91] without losing any phonetic interpretability as follows:

\[ [1] \rightarrow [+\text{fstress, +dstress}] \]
\[ [2] \text{ before [1]} \rightarrow [+\text{fstress, +dstress}] \]
\[ [2] \text{ after [1]} \rightarrow [−\text{fstress, +dstress}] \]
\[ [3] \rightarrow [−\text{fstress, +dstress}] \]
\[ [4] \rightarrow [−\text{fstress, +dstress}] \]
\[ [0] \rightarrow [−\text{fstress, −dstress}] \]

with ‘fstress’ denoting an accent associated with a major pitch movement and ‘dstress’ denoting an accent realized by durational lengthening of the syllable, and with + and − denoting the presence or absence of the specified feature.
Chapter 6

The Phono-Acoustical Model

6.1 Introduction

This chapter presents a brief overview of the phono-acoustical model of the SVOX system. The task of this system stage is to phonetically and acoustically interpret (or realize) the abstract, speaker-independent phonological representation that was explained in Chapter 5, that is, to map the symbolic phonological representation onto the continuous-valued speech signal. The main components of the phono-acoustical model are duration control, fundamental frequency control, and speech signal generation. The phono-acoustical model was designed and realized in large part by Hubert Kaeslin [Kae85, Kae86] and Karl Huber [Hub90, Hub91]. The author’s contributions lie in the design of fundamental frequency control, which will be discussed in detail in Chapter 7, and in implementing the TD-PSOLA method (cf. Section 6.6).

The first aim of the phono-acoustical model in the SVOX system was to mimic one specific speaker and one specific speaking-style as closely as possible. In contrast to the rule-based symbolic processing methods applied in the transcription stage of the SVOX system, the approaches taken for the phono-acoustical realization are based on statistical meth-
ods (duration and $F_0$ control) and on the direct combination of data extracted from natural speech (diphone synthesis). All these methods are in a certain sense “trainable” for their specific tasks, and they proved to be very effective for the mapping of symbolic data onto speaker-specific continuous physical data. Moreover, they are comparatively simpler to establish and use than rule-based methods.

Despite the fact that the SVOX system currently only produces one voice and one speaking style, the methods applied in the phonon-acoustical model could, of course, be used to realize other voices and speaking styles as well.

6.2 Segmental Phonology-to-Phonetics Mapping

Some symbolic processing is done at the beginning of the acoustic realization of the phonological representation. This processing should cover the phonology-to-phonetics mapping of speech segments (summarized as coarticulation in Chapter 2), which comprises a number of segmental reduction and assimilation phenomena, which depend on the speaking style and speaking rate. For example, the word combination [hat dːx], which contains a combination of two homorganic plosives, is reduced to [hɑːt] in fluent speech.

In the current SVOX system, however, the segmental phonology-to-phonetics mapping only comprises the reduction of unstressed long vowels and the substitution of certain phonemes, to account for the limited capabilities of the current speech generation component, as shown in the following sections.

6.2.1 Reduction of Long Vowels

According to the standard transcription given in [Dud74], vowels in many function words are long, e.g., “des”/[dɛs] or “fiir”/[fyːr]. This pronunciation is appropriate if these words are uttered in isolation but in the context of longer utterances, the vowels are often reduced in

length and in quality. For example, the above [fyːr] may become [fyː] or [fyː]

Although no systematic investigations were carried out on this subject, the following ad-hoc rule was incorporated in the SVOX system, which improved the fluency of the synthetic speech in many cases:

Long vowels are reduced to short vowels of the same quality if they are located in a completely unstressed syllable and if this syllable is in one of the following positions

... Boundary $RSy$ $RSy$ $AccSy$ ...
... Boundary $RSy$ $RSy$ AccSy ...
... Boundary $RSy$ AccSy ...

with

Boundary = word boundary or phrase boundary
$ = word boundary
AccSy = accented syllable (accent value ≠ 0)
$RSy$ = unaccented syllable where long vowels are shortened

This rule states that long vowels in unstressed syllables are converted to short vowels if the syllable does not lie at the end of a phrase and if the syllable is followed by at most one other unstressed syllable.

6.2.2 Phone Substitutions

Some phone substitutions are necessary to map the standard phonetic transcription onto the (more restricted) transcription that is needed to drive the diphone synthesis (see Section 6.3):

- The current speech signal generation (diphone synthesis) was designed for a Swiss usage of German only. The vocalic allophone of the /r/ phoneme and syllabic consonants were not used in the corresponding phonetic transcription. Therefore, the following substitutions are carried out:
6.3 Diphonization and Duration Control

Duration control and, closely connected with it, diphonization, map the phonological representation of an utterance (after the modifications described in Section 6.2) onto a sequence of basic synthesis units and corresponding duration values. In the case of the SVOX system, the synthesis units are diphones, i.e., phone transition elements (see Section 6.5).

Duration control and diphonization have been designed and implemented by Karl Huber, and they are described in detail in [Hub91, Hub90]. Duration control uses a statistical model, namely, a generalized linear model, to assign duration values to the synthesis units, based on several factors which are known to influence the duration of speech segments. These factors, which can all be extracted from the phonological representation of an utterance, include, among other things, the identities of the current segment and of its left and right neighbors, the position of the current segment in the syllable, in the foot (a rhythmical unit extending from one accented syllable to the next), in the phrase, and in the sentence, and the degree of accentuation of the current syllable. Duration control is schematically shown in Figure 6.1.

The coefficients used in the linear model were obtained from automatic statistical estimation procedures, in which the model was trained to optimally map phonological transcriptions of natural-speech utterances onto the corresponding duration data.

\[
\begin{align*}
|v| &\rightarrow |\text{or}|,\ |y| &\rightarrow |\text{u}|,\ |l| &\rightarrow |\text{al}|,\ |\text{n}| &\rightarrow |\text{ont}|,\ |\text{t}| &\rightarrow |\text{ont}|
\end{align*}
\]

- Affricates and diphthongs are treated like two separate sounds by the diphone synthesis. This is expressed by the substitutions

\[
\begin{align*}
|\text{au}| &\rightarrow |\text{au}|,\ |\text{ai}| &\rightarrow |\text{ai}|,\ |\text{yy}| &\rightarrow |\text{yy}|,\ |\text{ts}| &\rightarrow |\text{ts}|,\ |\text{tf}| &\rightarrow |\text{tf}|
\end{align*}
\]

- Foreign (mainly French) sounds are converted into German approximations:

\[
\begin{align*}
|\text{a}| &\rightarrow |\text{an}|,\ |\text{a}| &\rightarrow |\text{an}|,\ |\text{e}| &\rightarrow |\text{en}|,\ |\text{e}| &\rightarrow |\text{en}|,\ |\text{a}| &\rightarrow |\text{en}|,\ |\text{e}| &\rightarrow |\text{en}|,\ |\text{a}| &\rightarrow |\text{au}|,\ |\text{y}| &\rightarrow |\text{y}|,\ |\text{i}| &\rightarrow |\text{i}|,\ |\text{o}| &\rightarrow |\text{o}|
\end{align*}
\]
The actual realization of duration control is such that duration values are not predicted for diphone elements, but for triphones, where a triphone is an entire phone surrounded by two neighboring half-phones. These duration values of (overlapping) triphones are then converted into the duration values of diphones or semi-diphones. Semi-diphones are diphones split into two adjacent parts, when this splitting can be done reliably enough, as for instance in the case of voiced/unvoiced transition diphones. In the subsequent signal synthesis process, the units (semi-diphones or diphones) are stretched linearly in order to reach the specified duration values.

Besides the duration values of individual basic synthesis units, duration control also yields the time marks needed to synchronize $F_0$ generation with duration control. These time marks are the beginning, the middle, and the end of each syllable of the utterance. The middle of the syllable is defined as the time reached after the diphone that leads into the nucleus of the syllable (e.g., the time reached after the diphone [ba] in the syllable [baɪn]).

The following example illustrates input and output of the duration control process. Given the phonological representation


for the sentence "es regnet in Strømen" ("it’s pouring with rain"), the following unit/duration sequence is generated

\[ (/e, 76), (es\_36), (sr, 76), (re, 92), (\text{t}, 14), (e>, 40), \]
\[ (\text{g}_n, 50), (\text{g}_n, 22), (\text{a}, 46), (a>, 62), (\text{t}/, 45), (\text{t}/, 18), \]
\[ (/\text{a}, 48), (\text{m}, 40), (n\_s, 59), (n\_s, 43), (\text{f}/, 45), (\text{f}/, 21), (\text{t}, 77), \]
\[ (\text{r}, 79), (\text{t}, 15), (\text{om}, 74), (\text{m}, 45), (\text{a}, 67), (\text{n}/, 99) \]

Most units denote diphones, with ‘/’ denoting silence and '>' denoting the transition into a short pause. ‘\text{t}’ denotes sustainment of a sound. Some diphones are split into semi-diphones (indicated by the subscripts $a$ and $b$) in order to allow a finer control of the duration of the individual parts.

Duration control also generates the syllable times (in msec)

\[ (0.76, 194), (194, 338, 378), (378, 496, 609), (609, 675, 774), \]
\[ (774, 1054, 1091), (1091, 1173, 1339) \]

which are passed on to $F_0$ control.

### 6.4 Fundamental Frequency Control

The fundamental frequency ($F_0$) control component of the phonoaoustical model generates an $F_0$ contour based on the phonological representation of an utterance (after the modifications described in Section 6.2). In the SVOX system, this is achieved in a statistical manner and quite analogously to the method used for duration control, but instead of a trainable linear model, a non-linear model, i.e., an artificial neural network, is applied. As in the case of duration control, the neural network has been trained with natural-speech data in order to learn the mapping of phonological representations onto $F_0$ contours.

$F_0$ control is discussed in detail in Chapter 7. The result of $F_0$ control is a contour in the form of a simple list of time/value pairs. For the phonological representation given as an example in Section 6.3, the current $F_0$ control yields the contour

\[ (0.00, 117.5), (0.04, 117.5), (0.08, 116.1), (0.12, 113.7), \]
\[ (0.16, 110.8), (0.20, 110.4), (0.24, 111.0), (0.28, 113.2), \]
\[ (0.32, 113.9), (0.36, 113.3), (0.40, 112.8), (0.44, 112.2), \]
\[ (0.48, 110.7), (0.52, 106.4), (0.56, 101.8), (0.60, 98.7), \]
\[ (0.64, 95.9), (0.68, 91.1), (0.72, 86.7), (0.76, 84.3), \]
\[ (0.80, 82.6), (0.84, 88.8), (0.88, 91.9), (0.92, 94.1), \]
\[ (0.96, 95.3), (1.00, 95.8), (1.04, 92.8), (1.08, 87.1), \]
\[ (1.12, 88.6), (1.16, 76.8), (1.20, 74.6), (1.24, 73.1), \]
\[ (1.28, 72.6), (1.32, 72.8) \]

where the first value of each pair is the time in seconds, and the second value is the predicted $F_0$ value in Hertz. The time values are fully based on the syllable time marks (beginning, middle, and end) obtained from duration control. For the actual signal synthesis, $F_0$ values between the time points given in the contour are obtained from linear interpolations between neighboring contour points. (Of course, the predicted $F_0$ values are applicable only in voiced sections of synthesized speech signals; unvoiced segments remain unvoiced.)
6.5 Speech Signal Generation

The unit/duration list as output by the duration control component and the \( F_0 \) contour generated by the \( F_0 \) control component form the complete information that is needed to drive the actual speech signal synthesis of the SVOX system, which will be explained in this section.

6.5.1 Diphone Synthesis

Different approaches to the generation of synthetic speech signals have been presented in Section 2.2.1. The SVOX system applies diphone synthesis, which is currently the most commonly used concatenative synthesis scheme. Diphone synthesis consists of concatenating natural-speech phone transition elements (i.e., diphones) according to a given phone sequence, such that a new, synthetic utterance is constructed. Diphones cover the range from the middle of one phone to the middle of the next, thus preserving a high degree of naturalness in the perceptually very important spectral transitions from one speech sound to the next. For example, the word [haus] can be synthesized from the diphones [/h], [aus], and [us], which might be extracted from the natural-speech carrier words “hoch” [ho:x], “halten” [ba:ltın], “kaufen” [kaufın], “Muse” [mu:sa], and “Biss” [bis], respectively.

Two major problems arise with any concatenative synthesis approach: first, the extraction of basic synthesis units (such as diphones) from natural speech and the concatenation process should be such that the resulting spectral discontinuities at unit boundaries become minimal. Second, a method must be found to independently change the duration and pitch of the synthesis units without changing the spectral envelope (i.e., the shapes and positions of the formants). The following sections will briefly present the solutions to these problems adopted in the SVOX system.

6.5.2 Diphone Synthesis in the SVOX System

The diphone inventory used for signal synthesis in the SVOX system is due to Hubert KAESLIN [Kae85, Kae86]. The diphones were extracted from stressed syllables of polysyllabic German words wherever possible. The optimal diphone boundaries were obtained by means of so-called phone centroids, where a centroid was a typical, averaged spectrum of a specific speech sound, represented as a set of LPC (linear predictive coding) coefficients. The left and right boundaries of a diphone were set at the points of minimal spectral distance (the log-area distance) between the carrier word and the respective centroids. For example, to extract the diphone [jl] from the word “erschienen” [ɛʁʃi:nən], the distance contours between this carrier word and the centroids for the [J] and the [l] sound were measured, and the left and right diphone boundaries were (manually) set at the points of minimal distance to the [J] and the [l] centroid.

Phone centroids were established for all sustainable speech sounds. For diphones starting or ending in plosives ([p], [t], [k], [b], [d], and [g]), the respective diphone boundary was set at the beginning of the pre-plosive pause (i.e., at the beginning of the silent or near-silent interval preceding the sound burst).

In order to solve the problem of modifying pitch and duration of diphones, KAESLIN applied the LPC (linear predictive coding) method [Mak75, Kae85]. LPC, which was originally used as a data reduction method for speech signals, makes it possible to re-synthesize speech signals with different duration and at different pitch while preserving the original spectral shape of the signal. LPC therefore permits the easy manipulation of the prosodic parameters of concatenative speech synthesis units. However, this is achieved at the cost of a degradation of voice quality. Section 6.6 discusses the TD-PSOLA method, a newer and considerably better approach to prosody modifications in speech signals, which has been integrated in the SVOX system by the author in order to improve the quality of the synthetic speech. The SVOX system currently allows speech synthesis using either LPC or TD-PSOLA.
6.6 The TD-PSOLA Method for Prosody Modifications

6.6.1 Introduction

TD-PSOLA (time-domain pitch-synchronous overlap-add [CM89, MC90]) is a new method for the modification of pitch and duration of speech signals, which has gained much attention during the past few years and which has become a standard method for speech prosody manipulations.\(^1\) The method yields considerably better results than LPC, and, moreover, it is quite simple and requires only very little computation power (much less than LPC, for instance). The most critical point of the method is a careful labeling of the pitch periods of the original speech signal (individual periods of the pseudo-periodic parts of the signal, which correspond to single cycles of the vocal fold vibration). In applications such as speech synthesis, this labeling must be done only once for each new speech unit inventory, and if a good automatic labeling procedure can be used, the amount of preparatory processing becomes marginal.

The following sections further describe the method as it was implemented by the author in the SVOX system.

6.6.2 Preparatory Speech Signal Labeling

TD-PSOLA, as indicated by its name, operates fully in the time domain of the speech signal, i.e., on its waveform. The major prerequisite for its application is a labeling of unvoiced sections and of all periods of the speech signal in voiced parts, as shown in Figure 6.2.

All period labeling in the SVOX diphone signals was done manually by the author, based only on a speech waveform display. (For the 90 seconds of speech in the diphone inventory, this work took about three days, which is a small effort compared with the semi-automatic extraction of the diphones from carrier words.)\(^1\)

1The TD-PSOLA method has been patented by France Telecom, which prohibits its unlicensed use for commercial applications. For research purposes, however, the method can be applied freely.

![Figure 6.2: Marking of pitch pulses and labeling of pitch periods ('p') and unvoiced sections ('u') in the waveform of a part of the utterance “sechs” (“six”), as required for the application of TD-PSOLA.](image)

The unvoiced parts of the speech signal must be split into short pieces for the TD-PSOLA treatment. In the SVOX system, all unvoiced speech segments lasting longer than 10 msec were split up into pieces with a duration of 5 msec.

For the sake of simplicity, both the genuine periods of voiced signal parts and the short pieces of unvoiced signal parts will hereafter be called “periods”.

6.6.3 Signal Analysis

In order to modify the prosody of a speech signal, the signal is first analyzed as a sequence of short-term signals by successively extracting overlapping segments consisting of two adjacent (unvoiced or voiced) periods and multiplying these double periods with a window function (see Figure 6.3). Usually, a Hanning window (raised cosine function) is applied for this purpose, with the maximum of 1 at the pitch mark (beginning of a pitch period) and a decay to 0 toward the neighboring pitch marks. More precise specifications of the window function applied in the SVOX system will be given in Section 6.6.5.

In the present implementation, a double period is defined as being voiced or unvoiced depending on the voiced/unvoiced characteristics
of its right hand part. Thus, a double period at an unvoiced/voiced transition is treated as voiced, a double period at a voiced/unvoiced transition is treated as unvoiced.

The signal analysis need not be done in a separate step. In the present implementation, the analysis of a double period (extraction and application of the window function) is done just before it is needed in the resynthesis process.

6.6.4 Signal Resynthesis with Modified Prosody

Basically, the TD-PSOLA speech signal resynthesis consists of simply adding together short-term signals obtained from the analysis process. If all short-term signals are added together at their original locations and if an appropriate window function is applied, the original signal is completely reconstructed by the resynthesis process. Modifications of prosody can be achieved as follows:

- In order to raise or lower the fundamental frequency of voiced speech segments, the distance between pitch pulses (the centers of the double periods) is shortened or lengthened before adding the short-term signals together. In general, to set the fundamental frequency to the value \( f_0 \), the distance \( d \) between the pitch pulses must be set to \( d = [f_s/f_0 + 0.5] \) samples, where \( f_s \) is the sampling frequency of the signal.

  Pitch modifications also change the duration of the original signal. This must be compensated, if necessary, by corresponding duration modifications.

- **Duration modifications** are effected by omitting or repeating individual double periods, which leads to a shortening or lengthening of the original duration. The repetition of voiced double periods causes no problems. Unvoiced double periods, however, may not simply be repeated, because this leads to a strange voicing effect (at least if the unvoiced periods are of equal length). A feasible way to solve this problem is to simply time-reverse every odd repetition of an unvoiced double period (i.e., the first repetition is reversed, then the third, etc.). Using this procedure, duration stretches of unvoiced signal parts up to a factor of 2 do
not generate any undesirable sound effects, and the voicing effect arising from the application of higher stretching factors is reduced. For speech synthesis purposes, however, factors greater than 2 are hardly ever necessary.

Using the above scheme, duration modifications between factors of 0.5 and 2 are possible with hardly audible changes in the signal quality. \( F_0 \) changes are more problematic, but in the factor range 0.7 to 1.4 (lowering or raising of \( F_0 \) up to 6 semitones, which is about the range needed for speech synthesis), a fairly good quality of the speech signal can be maintained.

Modifications of duration and pitch can be carried out simultaneously in order to obtain the desired \( F_0 \) and duration values at every location in the speech signal. Figure 6.4 schematically summarizes these TD-PSOLA prosody modifications.

### 6.6.5 Implementation Details

**Window Function**

In the SVOX implementation of TD-PSOLA, the best signal quality was achieved using asymmetric Hanning-like windows. The window durations to the left and right of the current pitch pulse, \( W_l \) and \( W_r \), are defined as \( W_l = \min(T_l, T'_l) \) and \( W_r = \min(T_r, T'_r) \), where \( T_l \) and \( T_r \) are the original period durations of the left and right part of the double period, and \( T'_l \) and \( T'_r \) are the new period durations induced by the \( F_0 \) modifications. If the current pitch mark is set at time \( t = 0 \), the analysis window function therefore becomes

\[
w(t) = \begin{cases} 
\frac{1}{2}(1 + \cos(t \pi / T_{w})) & \text{for } -W_l \leq t \leq 0 \\
\frac{1}{2}(1 + \cos(t \pi / W_{w})) & \text{for } 0 \leq t \leq W_r \\
0 & \text{otherwise}
\end{cases}
\]

The minimization of the window lengths with the original period duration ensures that no part of the neighboring pitch pulses is included in the current double period, and the minimization with the new period lengths ensures that the influence of one double period does not extend into further periods.

![Figure 6.4: Duration and pitch changes of speech signals using the TD-PSOLA method. Double periods are schematically indicated by the analysis window function. The arrow indicates the original time axis, which is reversed in repeated unvoiced double periods.](image)

Original signal (double periods)

Raising pitch

Lowering pitch

Slowing down (double periods 2 and 4 unvoiced, 6 and 8 voiced)

Speeding up
In the SVOX system, asymmetric Hanning-like windows as defined above are usually applied. However, hardly any difference can be heard in the synthesized speech when triangular windows are applied instead. The advantage of triangular windows is that they can be computed much faster, i.e.,

\[ w(t) = \begin{cases} 
  1 + \frac{t}{W_n} & \text{for } -W_t \leq t \leq 0 \\
  1 - \frac{t}{W_r} & \text{for } 0 \leq t \leq W_r \\
  0 & \text{otherwise}
\end{cases} \]

The TD-PSOLA computations in the SVOX system are currently done in fixed-point arithmetic, which may considerably increase the synthesis performance on some computers. The fixed-point operations can be used safely and without loss of accuracy due to the small number of operations necessary on each sample, namely, one multiplication with the window function value and one addition with a sample of the (previous or next) overlapping double period.

**Treatment of Long Phones**

In certain cases in the SVOX diphone synthesis, it is necessary to sustain a sound for a rather long time. For example, the [f] and the [a] in [auf-fa-ran] are long phones. The diphone sequence generated for the whole word by the SVOX diphonization is [/a], [au], [uf][ff], [fa], [ta], [ar], [ra], [an], [n/], with [/] denoting the empty phone (silence), and with [ff] and [ta] denoting special sustaining units between diphones.

In the SVOX system, voiced sustaining units are realized by simply repeating the last period of the diphone preceding the sustaining unit as long as is necessary to account for the duration of the sustaining element. In the case of sustained unvoiced sounds, however, the sustaining elements are actually realized like diphones, i.e., as speech units extracted from natural speech.

The reason for the different treatment of voiced and unvoiced sustaining sounds lies in the fact that unvoiced speech signals should not be stretched by more than a factor of 2, and that, on the other hand, no more spectral gaps than are necessary (i.e., no more speech units than are necessary) should be used when synthesizing a speech signal.

**Period Types**

It is possible, in the SVOX implementation of TD-PSOLA diphone synthesis, to prescribe for each period (and thereby for the corresponding double period, of which the respective period forms the right hand part) whether it can be omitted, repeated or pitch-modified. Moreover, it is possible to assign a nominal fundamental frequency value to each period (instead of the regular \(F_0\) value computed from the period length). These special characteristics are all encoded as parts of the period label names.

Basically, all periods are labeled as either 'u' (unvoiced) or 'p' (voiced period). The following markers can be used to specify further properties:

- 'd': period cannot be deleted
- 'r': period cannot be repeated
- 'f': period cannot be pitch-modified
- <nn>: integer number to denote the nominal \(F_0\) value of a period

For example, the period label 'pr109' indicates that this is a voiced period which must not be repeated and which has a nominal \(F_0\) value of 109 Hz.

These special characterizations of periods might be useful in the future to achieve a better synthesis quality, especially in boundary conditions (voiced/unvoiced transitions, plosive bursts, and others). Up to now, however, none of these specifications have been used.
Chapter 7

Neural-Network-Based Fundamental Frequency Control

7.1 Introduction

The correct course of the fundamental frequency ($F_0$), that is, the pitch of the speech signal, is one of the most important factors for both naturalness and intelligibility of synthetic speech. Up to now, most of the research on TTS prosody has therefore been devoted to $F_0$ control.

In the standard German intonation, as described in [Ess56, Bie66], the $F_0$ contour is strongly related to accents and to phrase boundaries. Strong accents before the main accent of a phrase are marked in natural utterances by an $F_0$ rise or fall at a gradually decreasing general level. (Weaker accents are primarily marked by lengthening of the accented syllable.) After the main accent in a non-terminal phrase, $F_0$ stays at a high level and may even rise slightly towards the phrase boundary. After the boundary, $F_0$ starts at a low level. Within the syllable carrying the main accent of a terminal phrase (i.e., before the utterance end), $F_0$ drops to a low level and remains low thereafter. Figure 7.1 shows
the $F_0$ contour of the utterance “Touristikbulletin der schweizerischen Verkehrszentrale St. Gallen” (“tourist information report of the Swiss Central Tourist Office in St-Gall”) in which the main $F_0$ movements corresponding to phonological properties are marked.

Figure 7.1 also shows the gradual downdrift of the $F_0$ contour from the beginning towards the end of an utterance, which is known as declination. Declination is a universal phenomenon of spoken utterances [CC82]. In longer utterances, the declination is “reset” at certain phrase boundaries, that is, the general $F_0$ level is raised, and the declination downdrift starts again [Adr91, RM94].

The $F_0$ contour in Figure 7.1 is interrupted at unvoiced parts of the signal. In voiced parts, the $F_0$ contour shows micro-melodic patterns, which are caused by properties of the segments of which the utterance is composed. Examples of such influences are the facts that plosive bursts raise the $F_0$ value of a subsequent voicing onset, and that certain vowels tend to be realized at much higher $F_0$ values than others, i.e., they have a higher intrinsic pitch (e.g., the German vowels [y:] and [i:]).

By no means all $F_0$ movements in $F_0$ contours are perceptually relevant [Col90], and $F_0$ generators in TTS systems usually try to generate the important movements only. In German, these most important movements are the general declination behavior, and superimposed on it, rises and falls of $F_0$ at strongly accented syllables and at phrase boundaries.

$F_0$ contours for different languages have been characterized as movements that occur between a base and a top line, which limit the $F_0$ excursions [Adr91, Col91]. These lines both show a declining behavior with possible resetting points. This characterization is schematically shown in Figure 7.2.

Some authors prefer the concept of downstepping over declination, i.e., a downstepping of $F_0$ levels with each pitch accent instead of a constantly falling $F_0$ level that is independent of the accentuation pattern [Koh91].

$F_0$ generation in TTS systems is usually done in an explicit, rule-based manner (e.g., [Pie81, AHK87, Col91]). For German, an excellent rule system for $F_0$ generation has been established by Kohler [Koh90, Koh91]. A famous quantitative model for the description of $F_0$ contours of many languages has been presented by Öhman and Fujisaki [Öhm67, Fuj81]. In this model, $F_0$ contours are generated by means of impulse and step responses of second-order linear systems, which are driven by “accent” and “phrase” commands. An application of this model for German utterances has been described in [MPH93].

### 7.2 $F_0$ Generation in SVOX

Several methods have been tried for the generation of $F_0$ contours in the ETH TTS project. The most successful of these attempts were two implicit models of $F_0$ production, which were directly derived from a
corpus of natural-speech data. A *concatenative approach* has been described in [Tra90, Tra92], in which averaged natural F0 patterns were concatenated in such a way that they formed the F0 contour of new utterances according to their phonological representation. This method is very similar to the approach taken in [Aub90, Aub92]. The production of F0 contours from phonological representations by means of *recurrant neural networks* has also been reported in [Tra90, Tra92]. This neural-network-based approach to F0 generation proved to be the most successful, and it is the method currently applied in the SVOX system. The remainder of this chapter will be devoted to the description of this approach.

The basic idea of the application of neural networks to F0 synthesis was to train neural networks to imitate the human process of generating F0 contours from phonological representations. Instead of applying general rules to produce F0 contours, such neural networks should mimic one specific speaker and one specific speaking style only.

Other attempts to automatically learn to produce F0 contours from natural-speech data can be found in [LF86] (using hidden Markov models), [SG89, Sag90] (using neural networks, but in a different way than that presented in this chapter), and [EMC92] (statistical and symbolic learning procedures).

### 7.3 A Neural Network Architecture for F0 Prediction

Artificial neural networks, or simply *neural networks* (henceforth NN), are assemblies of relatively simple computational units (so-called *nodes*), which collectively perform an overall operation. NNs are realizations of the “parallel distributed processing” paradigm [RM86]. Good introductions to neural networks can be found in, e.g., [RM86, Lip87].

There are a number of highly different NN architectures, which are suitable for different tasks. In the NNs applied in the present report, the nodes typically sum up values coming from other nodes, perform a non-linear function on this input sum, and send out the resulting value to other nodes. The *links* between nodes are associated with a *weight*, by which the values transferred on them are multiplied, and which can be adjusted by *learning procedures* such that a specific task is fulfilled optimally. By *training* NNs through the use of an example set of input-to-output correspondences, an internal representation of this input-to-output mapping is derived, which is also expected to predict “reasonable” outputs for new inputs.

The idea of applying NNs to the mapping of phonological patterns onto F0 contours was originally inspired by the NETtalk system [SR86, SR87], in which a NN is used to predict the phonetic transcription of a sentence from its grapheme representation. The type of NN applied therein is a so-called *multi-layer perceptron* (MLP), in which the weights are adapted by *error backpropagation* [SR86, RHW86]. This network architecture is probably the most often used type for the mapping of input patterns onto output patterns.\(^1\)

The first concept of a NN for the prediction of F0 contours was to use the NN to predict F0 values relative to a base and a top line. Base and top lines would have had to be set up by a separate procedure. Moreover, it was intended that the F0 values be categorized into some few levels, such that the output of the NN would be used to select the most likely level category. The feasibility of such an approach for the prediction of artificial, rule-generated F0 contours was verified in [MK88]. However, the measurement of base and top lines in natural F0 contours and the prediction of these lines in TTS synthesis, which would have been necessary for the realization of this first concept, proved to be a great problem in itself.

The successful F0 algorithm applied in the CNET TTS system for French [LEM89], in which absolute F0 values from a natural-speech database were combined into synthetic F0 contours, then led to the idea of simply letting the NN predict the absolute course of F0 directly from the phonological representation of an utterance, by mapping continuous-valued outputs of the NN onto F0 values in an appropriate interval.

\(^1\)While the NETtalk system is of high relevance to studies of automatic and human learning abilities, the performance of NN-based methods for grapheme-to-phoneme mapping in practical TTS systems is far inferior to the application of dictionaries and purely symbolic computations, as has been pointed out in [CC190]. In the author’s personal view, a more promising place for the application of NNs in speech processing systems is the mapping of symbolic data onto statistical numeric data, or vice versa.
Figure 7.3: An Elman-type recurrent neural network with one hidden layer. Such networks were used for the task of predicting \(F_0\) contours from phonological representations. Some outputs of hidden nodes are fed back to the input layer upon the next step of an input sequence. At the beginning, the feedback input values must be provided explicitly. The “nodes” of the input layer (squares) simply distribute incoming values to the outgoing links, whereas the other nodes (circles) compute a sigmoid transfer function.

Since utterances can be of arbitrary length, only a part (a window) of the phonological representation should be presented to the network in order to predict a corresponding portion of the entire \(F_0\) contour. This seemed reasonable due to the fact that the relative course of \(F_0\) is influenced only by a small accentuation and phrasing context. However, some additional mechanism had to be used in order to handle the more general declination, which extends over rather long stretches in an utterance.

A recurrent network of Elman type [Elm90] was therefore applied for the prediction of \(F_0\) contours. Recurrent NNs of this type comprise a fully connected feed-forward part as in the case of multi-layer perceptrons, and, additionally, some internal node outputs emerging from one input presentation are stored and fed back to some special input nodes when the next input is presented. (These special nodes of the input layer are also called context units, since they provide the wider context of the current input.) The structure of such networks is depicted in Figure 7.3. Elman networks can be used to map entire sequences of inputs onto sequences of outputs, where the recurrent links serve as a “short-term memory”. According to [Lip87], MLPs with two hidden layers are (theoretically) able to handle more complex problems than MLPs with one hidden layer only. Experiments were therefore conducted with networks having one or two hidden layers, with feedback links from the first or second hidden layer to the input layer.

The NN architecture shown in Figure 7.3 was used to predict a portion of the entire \(F_0\) contour of an utterance from a portion of the phonological representation of this utterance. By shifting the input window through the entire input, the full \(F_0\) contour was obtained. The full contour simply consisted of the concatenation of all predicted partial contours. In the experiments described in this chapter, the output of one step always represented the \(F_0\) contour of a single syllable.

Preliminary tests with networks of the type presented in Figure 7.3 showed that these networks were well capable of memorizing individual \(F_0\) contours and that they could learn declination-like behavior. It could therefore be hoped that the networks would be capable of developing internal representations from examples of the phonology-to-\(F_0\) mapping and that they would predict appropriate \(F_0\) contours for new phonological representations.

All NNs used in the \(F_0\) prediction experiments had the following characteristics:

- The weights of all links between nodes were allowed to be arbitrary real values.
- Before the training all weights were initialized randomly with values between -0.3 and +0.3 (uniform distribution).
- As usual, input “nodes” (represented as squares in Figure 7.3) did not perform any specific transfer function but simply distributed input values to all outgoing links.
- All nodes with indices \(k\) of hidden layers and of the output layer performed the following sigmoid transfer function (shown in Fig-
\[ y_k = \frac{1}{1 + e^{-x_k}} = \frac{e^{x_k}}{e^{x_k} + 1} \]

where \( x_k \) is the weighted sum of input values to node \( k \), i.e.,

\[ x_k = \sum w_{ki} y_i \]

where \( w_{ki} \) is the weight of the link from node \( i \) to the current node \( k \), and \( y_i \) is the output value of node \( i \).

- As usual, an adaptable bias on the input of the sigmoid function (i.e., an adaptable shift of the transfer function with respect to the argument axis) was realized by a weighted link from every hidden and every output node to a dummy node in the previous layer, which constantly emitted the value 1.

### 7.4 The Training Corpus

#### 7.4.1 Text Material

Since neural networks learn from examples, the task of \( F_0 \) prediction from phonological representations required a corpus of natural \( F_0 \) contours together with corresponding phonological representations. Such a corpus of natural-speech data was established in the following way: a trained male speaker (the same speaker from whose voice the diphones used in the SVOX system had been extracted) read 186 sentences in a "neutral" news-reader style. The sentences were taken from 11 different texts (news, weather forecasts, tourist information), which had been extracted from newspapers. The sentences were of highly varying length: the number of syllables in a sentence ranged from 1 to 147, with an average of 35.4 syllables per sentence.

#### 7.4.2 \( F_0 \) Treatment

The sentences were LPC-analyzed, and \( F_0 \) values were computed every 15 ms using a pitch detection program developed by the ETH speech processing group. This was based on the autocorrelation function of the speech signal and applied a voicing decision adaptable to individual speakers. The raw computed \( F_0 \) contours were manually corrected and linearized. This piece-wise linearization very closely followed the original contour, but unvoiced signal sections were assigned virtual \( F_0 \) values by a linear interpolation between the neighboring voiced sections. Figure 7.5 shows an overlay plot of an original and a corrected and linearized \( F_0 \) contour. The piece-wise linearization, which was stored (like the original contour) in the form of equidistant \( F_0 \) samples (15 ms apart) was close enough to be perceptually indistinguishable from the original contour, if imposed on the LP-coded sentences.

The intention of the NN-based \( F_0 \) generation was to operate syllable-
wise, i.e., to produce an $F_0$ contour for one syllable in each step. This procedure required a constant amount of $F_0$ samples per syllable (corresponding to a constant number of NN output nodes). To this end, all sentences were manually segmented into demi-syllables, mainly by consulting the signal intensity contours and by listening to the resulting demi-syllable segments. The middle of the syllable, i.e., the separation between two demi-syllables, was set at the earliest intensity maximum in the syllable nucleus. Each of the demi-syllables was then again automatically split into two parts of equal length. For each of the resulting four segments of a syllable, the linearized $F_0$ contour was approximated by a linear regression line, and the $F_0$ values at the intersection points of the regression line with the segment boundaries were taken as $F_0$ values to be predicted by the NN, i.e., 8 time values and 8 corresponding $F_0$ values were stored for each syllable of an utterance. This extraction of $F_0$ values is schematically displayed in Figure 7.6.

The original $F_0$ contour of an utterance is very closely approximated by the 8 $F_0$ samples per syllable. It would be possible to indistinguishably approximate $F_0$ contours by fewer values (1 or 2 values, as for instance in [d’A93]) but the simple and “safe” method presented in this section was chosen in order not to lose any accuracy in any of the preparatory steps.

### 7.4.3 Phonological Transcription

Apart from $F_0$ values, which were the desired output of the NN for $F_0$ prediction, phonological representations of the sentences in the training corpus were needed as input to the NN. These representations were obtained by a manual transcription of the speech signal. This transcription was done only by the author in order to guarantee consistency over the entire corpus and in order to obtain a transcription which was similar to the kind of representation produced by accentuation and phrasing in the TTS process. (It would have been desirable to have been able to use the SVOX system to automatically generate the phonological representation of the sentences in the $F_0$ corpus, but this method was judged insufficiently reliable.)

For the segmental transcription, the standard phonetic transcription of the words in the sentences was used, rather than a close phonetic transcription of the uttered segments, because differences between the standard and a close phonetic transcription were of minor importance to the problem of $F_0$ prediction, and because a close segmental transcription would have required much more work.

Two types of phrase boundary were transcribed: one for pauses (denoted as $\#\{1\}$ in the phonological representation) and one for all other perceived sentence-internal phrase boundaries ($\#\{2\}$). $\#\{0\}$ was automatically assumed before and after each sentence.

Three types of accent were transcribed in a first step: pitch accents (accents associated with a major pitch movement), non-pitch accents on the main stress position in words, and secondary or tertiary word accents. All other syllables were judged unaccented. In a second step, this transcription was converted into the accent values used in the SVOX phonological representation as follows: The last pitch accent in each phrase was denoted as $[1]$, other pitch accents within the phrase as $[2]$. Non-pitch accents on the main stress positions of words in a phrase before $[1]$ were denoted as $[3]$, and after $[1]$ as either $[2]$ or $[3]$, depending on which value seemed appropriate in the context of the framework of the automatic accentuation procedure in the TTS process. Secondary and tertiary accents on words were denoted as $[4]$, and all unaccented syllables as $[0]$.

Some of the sentences of the $F_0$ corpus are given in Appendix C in
orthographic form together with the phonological transcription of the corresponding utterances.

### 7.5 Input and Output Coding

#### 7.5.1 Window Technique

As already mentioned, $F_0$ contours in the prediction experiments were generated for each syllable of an utterance. The $F_0$ contour of a syllable depends on a relatively wide phonological context as far as accentuation and phrasing information is concerned, whereas the influence of segmental properties on the $F_0$ contour of a syllable is much more local.

These considerations led to the following NN input treatment, which is depicted in Figure 7.7. From the phonological representation of an utterance, two different aligned streams of symbols were extracted. Each symbol in the first stream represented the accent value of a syllable, a phrase boundary, or a word boundary (denoted as $\#$). For each symbol of this first stream, a vector of binary features of segmental properties (i.e., a “segmental property symbol”) was put to the second stream. For accent symbols in the first stream, these properties represented segmental properties of the corresponding syllable. For boundary symbols in the first stream, some dummy feature vector was put to the second stream.

From each of the two streams, a window containing several symbols of the stream was extracted and, after a binary encoding, fed into the NN. The window in the first stream was usually larger than the window of the second. Of course, the symbol for which output had to be produced (the “focus” symbol) always had to be represented in both streams, i.e., the windows always overlapped by at least one symbol. By shifting both windows symbol by symbol and in parallel through both streams, the $F_0$ contour was generated for the entire phonological representation. For input symbols not representing a syllable (i.e., for boundary symbols), the NNs were trained to produce dummy $F_0$ values (cf. Section 7.5.3).

Due to the window technique applied to the input streams, some
filler symbols had to be set before and after both streams in order to enable the first and the last stream symbols to appear in the window focus position.

At the start of the computation of an entire $F_0$ contour, those input nodes of the NN that corresponded to feedback links had to be initialized with specific values. From the second input symbol presentation onward, these values were taken from the stored output values of hidden nodes of the previous input symbol presentation.

### 7.5.2 Input Coding

Different binary codings of the phonological information were used in the development of the NN-based $F_0$ control. For the training of the NN which is currently used in the SVOX system, the input coding was as follows:

**Stream 1:** (each symbol represented by 5 bits)

- $\#\{0\}$ starting a (P)-type phrase $\rightarrow 0, 1, 0, 0, 1$
- $\#\{0\}$ starting a (T)-type phrase $\rightarrow 0, 0, 0, 0, 1$
- $\#\{0\}$ at the end of an utterance $\rightarrow 0, 0, 1, 1, 1$
- $\#\{1\}$ starting a (P)-type phrase $\rightarrow 0, 1, 0, 1, 0$
- $\#\{2\}$ starting a (P)-type phrase $\rightarrow 0, 1, 0, 1, 1$
- $\#\{n\}$ with $n > 2$ treated like $\#\{2\}$

- $\$ (word boundary) $\rightarrow 0, 1, 1, 1, 1$

- 0 (accent) $\rightarrow 1, 0, 0, 0, 0$
- 1 (accent) $\rightarrow 1, 0, 0, 0, 1$
- 2 (accent) $\rightarrow 1, 0, 0, 1, 0$
- 3 (accent) $\rightarrow 1, 0, 0, 1, 1$
- 4 (accent) $\rightarrow 1, 0, 1, 0, 0$

- window filler symbol $\rightarrow 0, 0, 0, 0, 0$

**Stream 2:** (each symbol represented by 4 bits)

Each symbol was composed of the bits ‘ShortV’, ‘HighV’, ‘LUC’, ‘RUC’, with

- ShortV = 0 if syllable contains diphthong or long vowel (i.e., a vowel with lengthening symbol)
- ShortV = 1 otherwise
- HighV = 1 if the syllable nucleus was a vowel with generally high $F_0$ ([y], [i], [i], [u], [e], [e] in the $F_0$ corpus)
- HighV = 0 otherwise (including diphthongs)
- LUC = 0 if no consonant occurred to the left of the syllable nucleus or the consonant to the left of the nucleus was voiced and quasi-stationary ([l], [l], [m], [m], [a], [a], [r], [α], [s], [s], [i], [i], [u], [u])
- LUC = 1 otherwise
- RUC like LUC, but for the consonant to the right of the syllable nucleus

Bit settings for boundary symbols in the first stream: 0, 0, 0, 0

Bit settings for window filler symbols: 0, 0, 0, 0

**Feedbacks:** all feedback inputs were initialized to 0

All binary inputs to the NNs were realized as the real values 0.0 and 1.0 at the input nodes.

### Input Coding Example

From the phonological representation

```
#\{0\} (P) [1]mør-gan #\{2\} (T) [3]kømt [aɪn ga-] [v]t-tør #\{0\}
```

the two streams of symbols
\#\{0\} 1 0 #\{2\} 3 8 0 8 0 1 0 #\{0\} \\

and

\[
\begin{align*}
(0.0.0.0) & (0.0.0.0) (1.0.0.0) (1.0.1.0) (0.0.0.0) (1.0.1.0) (0.0.0.0) \\
(0.0.1.0) & (0.0.0.0) (1.0.1.0) (1.1.0.0) (1.0.1.0) (0.0.0.0) (0.0.0.0)
\end{align*}
\]

are extracted.

Assuming a window with 3 left and 6 right context symbols on the first input stream, and a window with 1 left and 1 right context symbol on the second, the binary coding of these streams (including the filler symbols) is

\[
\begin{align*}
(0.0.0.0.0) & (0.0.0.0.0) (0.0.0.0.0) (0.1.0.0.1) (1.0.0.0.1) (1.0.0.0.0) \\
(0.0.0.1.1) & (1.0.0.1.1) (0.1.1.1.1) (1.0.0.0.0) (0.1.1.1.1) (1.0.0.0.0) \\
(1.0.0.0.1) & (1.0.0.0.0) (0.0.1.1.1) (0.0.0.0.0) (0.0.0.0.0) (0.0.0.0.0)
\end{align*}
\]

and

\[
\begin{align*}
(0.0.0.0) & (0.0.0.0) (1.0.0.0) (1.0.1.0) (0.0.0.0) (1.0.1.0) (0.0.0.0) \\
(0.0.1.0) & (0.0.0.0) (1.0.1.0) (1.1.0.0) (1.0.1.0) (0.0.0.0) (0.0.0.0)
\end{align*}
\]

Assuming 10 feedback links, a NN for this configuration must therefore have

\[
\begin{align*}
n & = \text{numFeedbacks} \\
& + \text{SymLen1} \times (\text{WinLeft1} + \text{WinRight1} + 1) \\
& + \text{SymLen2} \times (\text{WinLeft2} + \text{WinRight2} + 1) \\
& = 10 + 5 \times (3 + 6 + 1) + 4 \times (1 + 1 + 1) \\
& = 72
\end{align*}
\]

input nodes.

For the first symbol applied to the NN (the initial boundary symbol), the full vector of 72 network input values (feedback initializations and binarized window contents of the first and the second stream) would therefore be

\[
\begin{align*}
0.0.0.0.0 \ldots 0.0.0.0.0 & 0.0.0.0.0 \ldots 0.1.0.0.1 \\
1.0.0.0.0 \ldots 1.0.0.0.0 & 0.0.0.1.1 \ldots 1.0.1.1.1 \ldots 1.0.0.0.0 \\
0.0.0.0 & 0.0.0.0.0 \ldots 1.0.0.0
\end{align*}
\]

7.6 NN Training

7.6.1 Training Algorithm

Feed-forward networks with nodes of the type discussed in this chapter are trained by error backpropagation [RHW86, Lip87]. In this well-known learning procedure, the deviations between desired output values and the output values actually produced by the network (i.e., the output errors) are propagated back to nodes in previous layers, and weights associated with links between nodes are changed according to their relative contribution to the error that occurs at the node at which the links end. The effect of these weight changes is a gradient optimization of the overall prediction error.

In order to train the recurrent networks used for \(F_0\) prediction, a variant of the error backpropagation scheme for sequences must be applied. This is described in e.g., [RHW86] and especially in [Wer90]. The backpropagation for sequences operates by treating the time axis as a spatial dimension: for a sequence of \(n\) input symbols and \(n\) corresponding output symbols, the recurrent network is regarded as being “unfolded” into a large, partially connected feed-forward network, in which the feed-forward part of the recurrent NN is repeated \(n\) times with identical weight matrices, and these \(n\) networks are connected by the recurrent links (see Figure 7.8).
Because in this view the recurrent links simply transfer node output values to the input layer of the next subnet, and because the nodes of this input layer simply distribute these values, the nodes with outgoing recurrent links in subnet \( k \) can be thought of as being directly connected to all nodes of the first hidden layer of subnet \( k + 1 \), with connection weights to be adapted during the training procedure.

The imaginary large network produces the entire sequence output from the entire sequence input in one feed-forward computation. Accordingly, the large network can be trained by error backpropagation from the output layer of the last sequence output to the input layer of the first sequence input.

In a practical implementation of a training step, the entire sequence with inputs \( i_1 \) to \( i_n \) is feed-forward computed using the same small network, and all feedback value vectors \( f_1 \) to \( f_{n-1} \) are stored, where \( f_k \) is the output vector of nodes with outgoing recurrent links in computation step \( k \). For the error backpropagation for the entire sequence from step \( n \) back to 1, the feed-forward computation with \( i_k \) and \( f_{k-1} \) at the input of the network in backward step \( k \) is done again, and the error between the produced output and the desired output of step \( k \) is propagated back to the input layer of the network (not only to the nodes of the first hidden layer as in the ordinary MLP backpropagation). The error signal thus produced at the input nodes must then simply be added to the error signal occurring at the nodes with outgoing recurrent links in step \( k - 1 \). The error signal at these nodes therefore consists of a part coming from the output error of step \( k - 1 \) and a part from the error propagated back to the input layer in step \( k \). Weight changes must be carried out according to the required change accumulated over the \( n \) backward steps.

In the author's first naive experiments on \( F_0 \) prediction with recurrent networks, the networks were not trained using the above "correct" training procedure, but simply by using the ordinary error backpropagation algorithm. The additional errors occurring at nodes with outgoing recurrent links were therefore simply ignored. This meant that errors were not propagated from the end of a sequence back to the beginning. In other words, the NN could access information from previous computations in the sequence which were done anyhow, but a specific element of a sequence could not influence the nature of these earlier computations. Nevertheless, as had been verified in preliminary tests,

**Figure 7.8:** Computation and training of a recurrent neural network by regarding it as "unfolded" into a large, partially connected feed-forward network with repeated weight matrices. The repetition factor corresponds to the length of the input/output sequence.
this training still allowed networks to learn to a certain degree sequential behavior such as declination, i.e., sequences of gradually decreasing values, which cannot simply be predicted from the input but require an internal memory.

The NN used in the current SVOX system, which is described in detail in [Tra95b], is one of the networks trained in this wrong mode since it has not yet been outperformed in its $F_0$ synthesis quality by newer and correctly trained networks. The SVOX network is a network with two hidden layers, with 20 nodes in the first and 10 nodes in the second hidden layer and 10 feedback links from the second hidden layer to the input layer. The input and output coding and window sizes are as described in Section 7.5. (It should be noted here that, unlike in the SVOX network, if networks with two hidden layers were trained with the correct backpropagation for sequences, feedbacks from the first hidden layer produced better results than feedbacks from the second hidden layer.)

Correctly trained networks which come very close in their performance to the SVOX network are networks with two hidden layers of about 20 and 10 nodes in the first and second hidden layers respectively, 10 feedback links from the first hidden layer to the input layer and with input and output coding and window sizes as described in Section 7.5. The difference to the SVOX network lies only in the fact that they produce slightly less lively $F_0$ contours.

Where not otherwise stated, the results provided in the subsequent sections were obtained by applying the correct backpropagation for sequences.

### 7.6.2 Training Experiments

In all experiments for $F_0$ prediction, the error at the output of the network was defined as the mean square error between the actual network output and the desired encoded $F_0$ values. A more perception-oriented error criterion would, of course, have been desirable but was not at hand. Usually, the first 100 sentences of the $F_0$ corpus were used as training material, and the remaining 86 for evaluation purposes. Different input and output coding schemes were used, but the coding described in Section 7.5 was applied to all networks for which results are reported in this thesis.

Figure 7.9 shows the course of the mean square error on the training and on the evaluation set (averaged over all training and evaluation examples, respectively) for a network with two hidden layers of 20 and 10 nodes, 10 feedback links from the first hidden layer to the input, and input and output coding and input window sizes as described in Section 7.5. The learning rate in this example was 0.02, i.e., the weight changes were 0.02 times the magnitude of the change given by the raw weight correction value of the backpropagation algorithm. The error course shown in Figure 7.9 is typical for such trainings: the error of the training examples continuously decreases, while the prediction error of the evaluation examples decreases in a first phase, reaches an optimum, and then increases again. The usual explanation for this behavior is that, in the first phase, the network learns to generalize and to develop an internal representation of the input-to-output mapping but in the second phase the network more and more simply memorizes the specific examples of the training set, and thereby loses some of the ability to generalize.
7.7 Results

Experiments were conducted with different input and output codings and with different configurations for the hidden layer(s) of NNs. The difficulty of choosing a good network for \( F_0 \) prediction lies in the fact that the simple mean square distance measure between natural and predicted \( F_0 \) contours is not a good indicator of the perceived naturalness of the generated contours. The networks had to be judged mainly by listening to synthesized \( F_0 \) contours.

Some of the best networks found in the experiments were networks with two hidden layers with about 20 nodes in the first and 10 nodes in the second hidden layer, 10 feedback links from the first hidden layer, input and output coding and input window sizes as described in Section 7.5, and trained on the first 100 sentences of the \( F_0 \) corpus. Slightly better acoustic results were achieved with two hidden layers than with only one, and for networks with two hidden layers, better results were obtained with feedback links from the first rather than from the second hidden layer.

Figure 7.10 displays the original linearized \( F_0 \) contours and NN-predicted contours of average quality for three of the sentences in the evaluation set of the \( F_0 \) corpus. (Of course, these predictions were not obtained from the full TTS synthesis, but started from the manual phonological transcription of the sentences.)

These networks produce quite acceptable \( F_0 \) contours, i.e., contours without disturbing “errors”. However, the contours are usually less lively than natural utterances. Some particular problems that occur with the networks are:

- Longer utterances consisting of several phrases usually sound much more acceptable than short one-phrase or one-word utterances. The reason for this might be that the training corpus was established with the aim of producing good \( F_0 \) contours especially for longer utterances and therefore contained many long sentences, which received more importance in the applied training procedure because of their higher number of syllables.
- Main phrase accents ([1]) on the last syllable of an utterance which should be realized as \( F_0 \) hills within this last syllable hardly show

![Figure 7.10: Original linearized (dotted lines) and smoothed NN-predicted \( F_0 \) contours (solid lines) for three different sentences of the \( F_0 \) evaluation set, obtained from a network with two hidden layers, with 20 nodes in the first and 10 nodes in the second hidden layer and 10 feedback links from the first hidden layer. The network was trained until the point of optimal prediction of all \( F_0 \) contours in the evaluation set. The prediction quality of the shown examples in terms of the mean square distance corresponds to the average prediction quality on the entire evaluation set of 86 sentences.](image)
any such $F_0$ movement. There seems to be a conflict between realizing the accent and realizing the usually low $F_0$ value on the last syllable, which is very constant for unstressed last syllables. The second rule seems to be stronger in this case.

- The $F_0$ rise in accented syllables with long vowels and, especially, diplhongs often starts earlier than in natural utterances.

[Tra95b] lists the weights of the network currently used for $F_0$ prediction in the SVOX system and presents the full algorithm for its application as an $F_0$ generator.

### 7.8 Formal Quality Evaluation

#### 7.8.1 Experiment

In order to evaluate the quality of $F_0$ prediction, two small formal listening experiments were conducted with naive subjects and with some members of the speech processing group. $F_0$ prediction in these experiments started from the manual phonological transcription of natural sentences, which means that the experiments tested the $F_0$ prediction alone, without the (possibly erroneous) automatic accentuation and phrasing procedure of the full TTS synthesis.

In the first experiment, the subjects had to listen to 40 pairs of LP-coded natural sentences (sampling rate 10 kHz). In each pair, one of the sentences carried the natural $F_0$ contour, the other carried the $F_0$ contour predicted by the NN currently used in the SVOX system. The natural and the synthetic version of the $F_0$ contour were presented in random order within each pair, and the subjects had to find out which one was the natural contour.

In the second, more difficult experiment, the subjects were presented 40 individual sentences which randomly carried either the natural or the synthetic $F_0$ contour predicted by the SVOX network. The subjects had to indicate whether they believed to have heard the natural or the synthetic contour.

The 80 test sentences were taken from the evaluation set of the $F_0$ corpus. These sentences were randomized, and the first 40 sentences of the randomized set were used for experiment 1, the second 40 sentences for experiment 2. In both experiments, the test sentences were presented in 4 groups of 10 stimuli (pairs or single sentences), with a short pause of about 1 minute between the groups. Each group was announced by a sound signal. A pause of 3 seconds was inserted after each stimulus, and a pause of 1 second between the two sentences of each pair in experiment 1. All stimuli were presented only once. The average duration of the 80 sentences of both experiments was 7.06 seconds, the minimum sentence duration was 0.54 seconds, the maximum duration 21.4 seconds.

Both experiments were carried out in one session. The total time for both experiments was approximately 20 minutes. In order to overcome some of the quality loss of the LP-coding, the stimuli were played back over a loud-speaker in a middle-sized room. The subjects were prepared for the experiment in the following manner: First, the subjects listened to some LP-coded sentences of a text in order to become familiar with the quality degradation of LP-coded speech. Second, different manipulations of the $F_0$ contour of two natural sentences were presented and explained in order to direct the subjects' attention to speech melody. Third, several pairs of sentences were presented as in the subsequent experiment 1, but always playing first the natural, then the synthetic version of the $F_0$ contour. (Of course, these preparatory pairs did not occur in the actual test.)

In the actual experiments, the subjects had to indicate their judgments on answer sheets which displayed the first word of each sentence.

#### 7.8.2 Results and Discussion

Both experiments were carried out with two groups of subjects. In both groups, the majority of the members were naive subjects (i.e., subjects not involved in speech processing or linguistics and with hardly any prior exposure to synthetic speech), but for comparison purposes the groups also included some members of the speech processing group. For the first group of subjects, sentences 1–40 of the randomized set of sentences were used in experiment 1, and sentences 41–80 in experiment 2. For the second group of subjects, the sentences used for experiment 1 and
2 were exchanged. In the following, only the results of the first group of subjects will be shown since the results for the second group turned out to be nearly identical.

Table 7.1 shows the “recognition rates” of the individual subjects on both experiments (i.e., the number of correct answers divided by the total number of answers in per cent) and the average recognition rates. Table 7.2 shows the results if only long sentences are considered (i.e., sentences with a duration above the average of 7.06 seconds, which is the case for nearly half of the sentences). The experiments were designed such that completely random answers would result in a recognition rate of 50%.

The first conclusion that must be drawn from the results given in Tables 7.1 and 7.2 is that the ability to judge the naturalness of the

<table>
<thead>
<tr>
<th>Subject</th>
<th>Naive</th>
<th>Recogn. Rate Exper. 1 [%]</th>
<th>Recogn. Rate Exper. 2 [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>70.0</td>
<td>75.0</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>72.5</td>
<td>67.5</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>72.5</td>
<td>65.0</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>70.0</td>
<td>67.5</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>57.5</td>
<td>52.5</td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>85.0</td>
<td>72.5</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>70.0</td>
<td>62.5</td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>82.5</td>
<td>80.0</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>82.5</td>
<td>80.0</td>
</tr>
<tr>
<td>average 1-9</td>
<td></td>
<td>73.6</td>
<td>69.7</td>
</tr>
<tr>
<td>10</td>
<td>no</td>
<td>92.5</td>
<td>80.0</td>
</tr>
<tr>
<td>11</td>
<td>no</td>
<td>52.5</td>
<td>50.0</td>
</tr>
<tr>
<td>12</td>
<td>no</td>
<td>65.0</td>
<td>55.0</td>
</tr>
<tr>
<td>average 10-12</td>
<td></td>
<td>70.0</td>
<td>61.7</td>
</tr>
</tbody>
</table>

Table 7.2: Recognition rates in two F₀ evaluation experiments when only sentences with a duration above average (i.e., longer than 7.06 seconds) are considered.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Naive</th>
<th>Recogn. Rate Exper. 1 [%]</th>
<th>Recogn. Rate Exper. 2 [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>64.7</td>
<td>75.0</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>76.5</td>
<td>80.0</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>70.6</td>
<td>65.0</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>64.7</td>
<td>65.0</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>64.7</td>
<td>65.0</td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>82.4</td>
<td>75.0</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>70.6</td>
<td>60.0</td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>88.2</td>
<td>80.0</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>82.4</td>
<td>85.0</td>
</tr>
<tr>
<td>average 1-9</td>
<td></td>
<td>73.9</td>
<td>72.2</td>
</tr>
<tr>
<td>10</td>
<td>no</td>
<td>100.0</td>
<td>85.0</td>
</tr>
<tr>
<td>11</td>
<td>no</td>
<td>47.1</td>
<td>35.0</td>
</tr>
<tr>
<td>12</td>
<td>no</td>
<td>64.7</td>
<td>60.0</td>
</tr>
<tr>
<td>average 10-12</td>
<td></td>
<td>70.6</td>
<td>60.0</td>
</tr>
</tbody>
</table>

The melody of sentences in the described experimental setup varies considerably between different subjects, even between the members of the speech processing group and even if the synthetic F₀ contour is presented together with the natural contour. Therefore, only the best results should be used for a judgment of the quality of the F₀ prediction.

If random answers are assumed in experiment 1 in cases where the synthetic contours appear completely natural, the best naive recognition rate of 85% in Table 1 suggests that about 2 * (100 - 85) % = 30% of the synthetic contours sound as natural as to be confused with natural contours if presented to naive listeners. If presented to a trained listener (subject 10), this is the case for about 15% of all sentences, and for none of the long sentences (Table 7.2). As expected, experiment 2 proved to be more difficult, and the best results for experiment 2 in Tables 7.1 and 7.2 (85%) suggest that about 30% of all synthetic F₀ contours sound highly natural if presented in isolation.

These results show that it is possible to produce highly natural F₀
contours by means of the NN method presented in this chapter, based on rather restricted phonological information. However, the results also show that much remains to be done in order to obtain completely natural-sounding F0 even for the limited pragmatic context of neutral information communication.

### 7.9 Generalization Behavior

It is difficult to investigate what rules a NN has actually learned. Looking at weight values seems nearly hopeless. The Figures 7.11...7.15 therefore simply present NN responses to some rather special, unnatural inputs, which might serve as indicators of the internal rule representation that may be developed in networks of the type presented in this chapter. All responses are produced by a network with one hidden layer of 30 nodes, 10 feedback links, the usual input and output coding and input window sizes, and trained until the point of optimal prediction on the evaluation set. (This point in training is, however, not the best as far as the overall acoustic impression of this NN is concerned.)

Most of the figures are given with a syllable count abscissa, i.e., one unit corresponds to one syllable. Where not otherwise stated, the syllable properties were constantly set to those of an input syllable [a:].

Despite the “correct” generalizations shown in Figures 7.11...7.15, many F0 patterns produced by trained NNs still seem to be attributed to specific input patterns only, and, in general, the networks seem to have learned fewer useful generalizations than might be suspected from the given examples. This is not surprising if one considers the fact that the input space (represented by 72 bits) is extremely sparsely filled by the few training examples. Future research might attempt to compress the input space in order to force NNs to learn more abstract concepts.

### 7.10 F0 Postprocessing

The raw F0 outputs from NNs are somewhat ragged curves. In the SVOX system, the F0 contours are therefore slightly smoothed before they are applied to the synthetic signal. In order to do so, the network

![Figure 7.11: F0 contour produced for the accentuation and phrasing input #0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #0, i.e., for a sequence of completely unstressed syllables in one phrase. The resulting F0 contour resembles a “declination base line”. The contour indicates that the recurrent links are actually used, since the declination occurs even in the input range where the beginning and end of the utterance are not visible in the input window, and the input remains constant from one syllable to the next (i.e., after the third syllable and before the last syllable but size).](image1)

![Figure 7.12: F0 contour produced for the accentuation and phrasing pattern #0 0 0 0 0 0 0 0 2 0 0 0 0 0 2 0 0 0 0 0 #0, i.e., a sequence of unstressed syllables and some syllables with accent [2]. The contour is actually a downstepping pattern of F0 hills of the type that would be expected in natural speech.](image2)
Figure 7.13: “Declination resetting” in the $F_0$ contour produced for the accentuation and phrasing pattern $\neq \{0\} 0 0 0 0 0 0 0 0 0 0 0 \neq \{1\} 0 0 0 0 0 0 0 \neq \{0\}$. i.e., a sequence of unstressed syllables interrupted by a phrase boundary. The “declination line” is reset to a slightly higher level after the boundary.

Figure 7.14: Overlay plot of two $F_0$ contours produced for the same accentuation and phrasing pattern $\neq \{0\} 0 0 0 0 0 1 0 0 0 0 0 \neq \{0\}$, but once with the accented syllable [1] declared to contain a vowel of low intrinsic pitch (solid line) and once a vowel of high intrinsic pitch (dotted line). As expected, the vowel with high intrinsic pitch produces a larger $F_0$ excursion.

Figure 7.15: Overlay plot of two $F_0$ contours predicted for the same corpus sentence of the evaluation set, once with the feedbacks enabled (solid line) and once with the feedback values artificially set to their initialization values for each sequence step (dotted line). The difference in the contours suggests that the recurrent links are mostly used to control the general level of $F_0$ whereas the more local phenomena are controlled by the direct input to the network.

Figure 7.16: Raw $F_0$ contour as produced at the output of the neural network (dotted line) and the $F_0$ contour after a slight smoothing (solid line). The smooth contour is used in the actual signal synthesis.
output $F_0$ contours (with 8 time/value pairs per syllable) are sampled at constant intervals of 40 ms, and a moving average filter with window size 3 is applied once to the sampled contour. Figure 7.16 displays an example of the raw NN-generated contour and the $F_0$ contour after smoothing.

7.11 Conclusions

The best $F_0$ generator for the SVOX system was realized by using a recurrent neural network as $F_0$ predictor. This approach required only a limited human effort in order to obtain an $F_0$ generator of high quality. It was much more important to know what phonological properties control the phonetic realization of intonation than how these properties are realized quantitatively. The qualitative part of the task of building an $F_0$ predictor was therefore based on well-known linguistic (especially phonetic) research results. The quantitative part, however, was derived automatically by optimizing the parameters of a non-linear model for the prediction of $F_0$ contours, based on given statistical data. The advantage of this approach lies in the fact that a high quality of $F_0$ generation was reached with only little research on natural $F_0$ production. Disadvantages of this approach are that little insight is gained into the actual rules that govern the phonetic-acoustic realization of intonation, and, connected with this problem, that it is rather difficult to effect a given desired change of the behavior of the resulting $F_0$ generator.

One of the most useful prerequisites for future research on $F_0$ prediction by means of neural networks would be a perceptually appropriate, computable measure of the deviation of synthetic contours from natural contours. Such a measure could be used for the training as well as for the evaluation of $F_0$ predictors.

The author has experienced many objections and skeptical comments concerning the application of neural networks. They were for instance entitled “black magic”. It would in fact be black magic if a neural network would predict $F_0$ contours from, e.g., orthographic input text. However, since the symbolic phonological data (accents, phrase boundaries) and the physical data ($F_0$ contour) are rather tightly coupled, the NN training plays the role of an estimation procedure for the parameters of a non-linear statistical model. Such statistical parameters occur in any $F_0$ predictor, even in rule-based systems. The question is therefore not if, but how far the statistical method should go, i.e., how closely related the input of the statistical model should be to the $F_0$ contour. In the case of the direct prediction of $F_0$ from accents and phrase boundaries, as in the SVOX system, the statistical method admittedly goes very far. In future, there might well intervene an intermediate rule-based step for the prediction of a symbolic representation of the intonation from accents and phrase boundaries. The statistical method would then have to map a symbolic description of the intonation of an utterance onto physical $F_0$ values, which could possibly be done even better than the mapping described in this chapter.
Chapter 8

Summary: A Simple Text-to-Speech System

The first aim of the ETH TTS project was not to build a small real-time system (which should fit within 640kByte of computer memory, as in many other projects). The aim has rather been to build a linguistically motivated system using general approaches to the different problems occurring in TTS synthesis. This led to a prototype system of tremendous size. The SVOX system presented in this thesis is still very large due to the fact that it is still more a research instrument than a "naked" TTS program. Despite its size, the SVOX system consists of relatively simple building blocks, which have been presented in this report and which will be summarized briefly in the following sections.

Some TTS projects have attempted simplicity by applying the same formalism (such as, e.g., the Delta rule development system [HKK85]) all over the entire synthesis system. The ETH TTS project developed along a different line: the basic philosophy was to apply solutions which seemed most appropriate and promising for the individual partial problems. This approach is most obviously expressed in the different natures of the methods used in the transcription and in the phono-acoustical model.

The SVOX TTS system starts with a morphological and syntactic
analysis of input words and sentences, as explained in Chapter 3. This analysis was implemented in the concise form of definite clause grammars, which are based on production rule systems. Such production rule grammars are well-known to both linguists and computer scientists. (The fact that the formalism is simple and easy to understand does not, of course, imply that it is easy to write a grammar for a natural language. However, a concise formalism such as DCGs may help to avoid unnecessary complications of this task.) The two-level rules, which work in connection with the morphological analysis, serve to keep the lexicons and the word grammar small and general.

Chapter 4 showed how abbreviations, numbers, and novel words can be converted into pronunciation form in the general framework of the morpho-syntactic analysis, without extending the formalism and without adding new parsing methods. Whereas other systems start by an explicit preprocessing stage with separate conversion procedures for these special grapheme-to-phoneme mapping problems, the SVOX system integrates the entire grapheme-to-phoneme mapping within the "ordinary" morpho-syntactic analysis. The special grapheme-to-phoneme mapping of abbreviations, numbers, and novel words could therefore be realized as a "by-product" of the morpho-syntactic analysis.

Accentuation and prosodic phrasing, presented in Chapter 5, are strongly based on works in generative phonology. Accentuation is mainly based on the nuclear stress rule, which reduces non-nuclear stresses within each constituent, and on a rhythmic stress shift rule. The phrasing algorithm builds larger phrases from small temporary phrases according to the syntactic structure of a sentence and according to a criterion for the minimum length of an independent phrase, based on accent and syllable counts. The accentuation and prosodic phrasing scheme are probably the most language-specific and least general parts of the SVOX system. Unlike for the other parts of the synthesis system, it is not clear if and to what extent these methods could be applied to languages other than German.

The phono-acoustical model described in Chapters 6 and 7 interprets the phonological representation of an utterance in terms of phonetics and acoustics. Other TTS systems, e.g., the MITalk system [AHK87] or the KTH system [CG76], apply sophisticated explicit rule systems for this interpretation, which required many years of research to establish. In the SVOX system, implicit, trainable methods, which directly map the phonological information onto acoustic parameters, are applied for the same purpose. For duration control, a generalized linear model is used, and \( F_0 \) control is achieved by applying a neural network. The synthetic speech signal is generated by the concatenation of diphone elements which have been extracted from natural speech, and the prosody parameters are imposed on the synthetic speech signal by using the simple but very effective TD-PSOLA method. The implicit approaches in the phono-acoustical model proved to be very successful and probably required less effort than the establishment of explicit rule systems. The major drawbacks of these approaches are that they are less flexible than explicit rule systems and that they do not provide much insight into the underlying phonetics.

Most parts of the SVOX system are based on very general and language-independent methods. Ad-hoc solutions can currently be found in the syllabification procedure (Chapter 5), the coarticulation rules (segmental phonology-to-phonetics mapping, Chapter 6), and in some parts of the accentuation and prosodic phrasing algorithms (Chapter 5).

The SVOX system is currently regarded as one of the best TTS systems for German. But, of course, all parts of the system could be further improved, especially as far as the linguistic knowledge bases incorporated in the system are concerned. Apart from such improvements to the TTS synthesis of German, a future application of the SVOX system to another language could reveal the generality and the limitations of the SVOX architecture.
Appendix A

ASCII-Representation of German IPA Symbols

The SVOX system uses an ASCII representation of the IPA (International Phonetic Association) phonetic symbols used in the German language. These ASCII forms frequently occur in the main part of this thesis. The following table displays the IPA symbols, the corresponding ASCII coding and the graphemic and phonetic representation of a word that contains the corresponding phone, as given in [Dud74, p. 14].

Usually, upper- and lowercase letters in the ASCII coding of the IPA symbols are treated as equal in the SVOX system. In the lexicons of Chapters 3 and 4, however, upper- and lowercase letters are distinguished: Lowercase letters are used for ordinary IPA symbols, and uppercase letters represent special symbols used, for instance, in the two-level rules.

<table>
<thead>
<tr>
<th>IPA</th>
<th>ASCII</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>hat</td>
</tr>
<tr>
<td>a:</td>
<td>a;</td>
<td>Bahn</td>
</tr>
<tr>
<td>u</td>
<td>4</td>
<td>Ober</td>
</tr>
<tr>
<td>y</td>
<td>~4</td>
<td>Uhr</td>
</tr>
<tr>
<td>á</td>
<td>~a</td>
<td>Pensee</td>
</tr>
</tbody>
</table>
ä: ~a; Gourmand  [gʊɐ̯mˈmɑː]
ai a.1 weit  ['vaɪt]
au a.u Haut  ['haut]
b b Ball  ['bal]
c c ich  ['ɪŋ]
d d dann  ['dɑn]
dʒ d.ʒ Gin  ['dʒɪn]
e e Methan  [ˈmeθa:n]
ei e; Beet  ['bɛt]
ɛ 3 hätte  [ˈhɛtə]
ɛː 3; wähle  [ˈvɐː.lɛ]
ɛː ~3; timbrierten  [tinˈbɾɪʁənt]
ɛː: ~3; Timbre  [tinˈbruː]
ø 6 halte  [ˈhɑltə]
ʃ f Fass  ['fa:s]
ɡ g Gast  [ɡaːst]
h h hat  [hɑt]
i i 1; ei; viel  [ˈviːl]
iː i Stude [ˈʃtud ip]
iː bist  [ˈbɪst]
j j ja  [ˈjaː]
k k kalt  [ˈkɑlt]
l l Last  [ˈlɑst]
ˈ 1 Nabel  [ˈnaːbəl]
m m Mast  [ˈmɑst]
ŋ 3,m grossen  [ˈɡɹʊʃn]
n n Naht  [ˈnaːt]
ŋ 3,n baden  [ˈbɑdən]
ŋ 3 9 lang  [ˈlɑŋ]
o o Moral  [ˈmoːrəl]
oː o; Boot  [ˈboot]
ø 3,ø; loyal  [ˈloːjəl]
œː ~ø; Fondue  [ˈfɔ̃dyː]
œː ~ø; Fond  [ˈfɔ̃d]
ʊ q Post  [ˈpɔst]
ʊ 0 Økonom  [ɔkɔˈnɔm]
ʊː 0; Öl  [ˈoːl]
œ 8 göttlich  [ˈɡɔtliχ]
œː ~8; Lundist  [ˈlʊntɪst]
Appendix B

Formalisms in the Morpho-Syntactic Analysis

This appendix presents the exact definitions of the format of all knowledge bases used in the SVOX morpho-syntactic analysis.

Basic Symbols

AnyChar ::= ... (any character >= ',').
Letter ::= "A" | ... | "Z" | "a" | ... | "z".
Digit ::= "0" | ... | "9".
Special ::= ";" | ";" | ";".
AlphaNum ::= Letter | Digit | Special.
Character ::= "" AnyChar.
NullCharacter ::= ".".
Appendix B. Formalisms in Morpho-Syntactic Analysis

Ident ::= Letter {AlphaNum}.
VarIdent ::= "?" {AlphaNum}.
String ::= "", {AnyChar} ".".
Number ::= [ "-" | "+" ] Digit {Digit}.

Remarks

- Basic symbols may not contain any blanks except where 'AnyChar' occurs.
- Upper- and lowercase characters are equivalent in identifiers.

Comments

LineComment ::= "!" {AnyCharButEOL} EOL.
RangeComment ::= "[" {AnyCharButSqBrackets | RangeComment} "]".

Remarks

- Comments may be used like blanks between basic symbols.
- Strings are delimited by double quotes; to include double quotes in a string, "." must be used for each double quote character.
- Within a comment started by "!", all characters on the same line are ignored, including '!' and ';' i.e., a range comment may not be started within a line comment, unless it is finished on the same line.
- Within a range comment of any nesting depth, all characters except '!' and ';' are ignored, including ';'; i.e., a line comment may not be used to delete a range comment start or a range comment end.

Common Definitions

Atom ::= Ident.
Variable ::= VarIdent.
TermTuple ::= "(" [Term ("," Term)] ")".
Term ::= Atom | Variable | TermTuple.
Quality ::= Number.

ConsDeclaration ::= "::CONS" ConsName [ConsDescr] [Features] [AddFeatures] ":".
Features ::= ":FEAT" Feature {Feature}.
Feature ::= FeatureName [FeatureDescr] [AllowedValues].
AllowedValues ::= ":" [Value [ValueDescr] ":"].
AddFeatures ::= ":AND" FeatValPair [FeatValPair].
FeatValPair ::= "<" FeatureName Term ">".
ConsName ::= Ident.
ConsDescr ::= String.
FeatureName ::= Ident.
FeatureDescr ::= String.
Value ::= Atom.
ValueDescr ::= String.

ConsAndFeatures ::= ConsName [TermTuple].

Remarks
• The empty feature list can be expressed by ‘O’ or by completely omitting the feature list parentheses.

Syntax of Lexicon Description Files

LexionDescr ::= {ConsDeclaration}.

Remarks

• Lexicon description files should be used to store permanent comments and constituent declarations. Since lexicon files may be created automatically (without any comments), these optional files provide a means to prevent documentation comments from being overwritten. Currently, there may be one description file associated with each lexicon.

Syntax of Lexicons

Lexicon ::= {ConsDeclaration | LexEntry}.

LexEntry ::= ConsAndFeatures GraphString PhonString [Quality].

GraphString ::= String.

PhonString ::= String.

Remarks

• A lexicon consists of any number of lexicon entries and constituent declarations. Each lexicon entry consists of a constituent name and a feature term tuple, the graphemic string and a corresponding phonemic string.

• If the quality value is omitted, a minimal penalty value (= 1) is assumed.

Syntax of Grammars

Grammar ::= (ConsDeclaration | GrammarRule).

GrammarRule ::= Head "==>" Tail "*" [Quality] [Invisibility].

Head ::= ConsAndFeatures.

Tail ::= {ConsAndFeatures}.

Invisibility ::= ":INV".

Remarks

• A grammar consists of production rules in DCG-like form and constituent declarations. Each production rule consists of a head (a constituent with a tuple of features) and a (possibly empty) tail, which is a list of subconstituents with features. Each rule is terminated by a quality number, which is added to the overall quality of a syntactic derivation when the rule is applied, and possibly an invisibility marker, which determines whether the rule will produce a node in the parse tree or not.

• If the quality value is omitted, a minimal penalty value (= 1) is assumed.

Syntax of Two-Level Rules

TwoLevelRules ::= [Alphabet] {Automaton}.

Alphabet ::= "::ALPHABET" {LexSurfPair} "*".

LexSurfPair ::= LexChar ["/" SurfChar].

LexChar ::= Character | NullCharacter.

SurfChar ::= Character | NullCharacter.

Automaton ::= "::AUTOMATON" AutoName InputClasses Transitions "*".
Syntax of System Parameter Collections

SysParams := \{ ParameterSetting \}.

ParameterSetting := ":PARAM" ParamName ParamValue.

ParamName := Ident.

ParamValue := String | Number | ":TRUE" | ":FALSE".

Syntax of Lexicon Editor Descriptions

LexEdDescription := Grammar.

Lexicon editor descriptions basically obey the same syntax as grammars, but all grammar rules must be of the form:

EDCONS("<ConstituentName>,"<ConstituentDescr>) => ...  

where "ConstituentName" refers to the lexicon constituent to be edited, and "ConstituentDescr" is an atom which describes the constituent and which may be used to subcategorize the main constituent. The body of the grammar rule usually contains the main constituent and possibly additional constituents (like inflection endings). Each rule (interpreted as generating grammar rule) is used to produce an entry in the word form display field of the editor (in the order of appearance in the editor description file).

Current restrictions:

• The body of a grammar rule must either be empty or consist of the constituent to be edited followed by one additional constituent (usually an inflection constituent).

• All features of the constituent to edit must be variable (the minimum number of features is 1), and exactly one of these variables must occur in the right-hand additional constituent; the remaining features of the additional constituent may be variables or atoms.
• All rules belonging to the same constituent to edit must occur together in one sequence, and also all additional constituents of the same type must occur together in sequence.

Example

The following code defines the editing of the constituent ‘NS’ (noun stem). The constituent declaration is not necessary if defined elsewhere (e.g., in the morpheme lexicon or in a lexicon description file). To allow the user to check the appropriate selection of the singular and plural inflection category ('SC' and 'PC'), a display of full forms consisting of the edited noun stem followed by the inflection endings (nominative, genitive, dative, and accusative case) will be produced.

```plaintext
:CONS NS "Noun stem"
:FEATURES SC "Sing.cl." {SK0 SK1 SK2 SK3 SK4 SK5 SK6
SK7 SK8 SK9 SK10 SK11}
P C "Plur.cl." {PK0 PK1 PK2 PK3 PK4 PK5 PK6
PK7 PK8 PK9 PK10 PK11}
G "Gender" {F "Fem" M "Mas" N "Neut"}
S "Surf.cat." {N "Noun"}

EDCONS(NS,NOUN_STEM) == NS (?SC, ?PC, ?G, ?S) NES (?SC,D,S3) *
EDCONS(NS,NOUN_STEM) == NS (?SC, ?PC, ?G, ?S) NEP (?PC,N,P3) *
EDCONS(NS,NOUN_STEM) == NS (?SC, ?PC, ?G, ?S) NEP (?PC,D,P3) *

EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S1,PRES) *
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S2,PRES) *
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S3,PRES) *
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S1,PAST) *

EDCONS(VS,Past_only_stem) ==> VS(?INF,?TYPE) VE(?INF,S1,PAST) *
EDCONS(VS,Past_only_stem) ==> VS(?INF,?TYPE) VE(?INF,S2,PAST) *
EDCONS(VS,Past_only_stem) ==> VS(?INF,?TYPE) VE(?INF,S3,PAST) *
```

allows to edit verb stems VS of two different types separately, although they belong to the same morpheme category. The first stem type is the regular verb stem, for which some present tense and past tense inflection forms are displayed, and the second type is the past tense form of an irregular verb stem, which is shown with past-tense inflection forms only.

If the constituent description atom (second feature in ‘EDCONS’ head) changes in the list of production rules, a new editable category will be started. For example,

```plaintext
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S1,PRES) *
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S2,PRES) *
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S3,PRES) *
EDCONS(VS,Regular_ Stem) ==> VS(?INF,?TYPE) VE(?INF,S1,PAST) *
```
Appendix C

F\textsubscript{0} Network Training Corpus

186 sentences from 11 texts were used to train and test the neural networks for F\textsubscript{0} control. Some of these sentences (1..10 and 181..186) are given below in orthographic and phonological form.

The segmental transcription in the phonological representation (given in ASCII form as described in Appendix A) is not a close phonetic transcription of the actual utterances, but the standard form according to [Dud74] (except for [i], [u], [y], and [w], which are replaced by [ə], [ʌ], [ɔ], and [ɪ], respectively). The glottal stops before word-initial vowels are not included in the transcription.

001
Friedliche Massenkundgebung in Peking


002
Beteiligung von Passanten

003
Peking, 27. April.

004
Trotz massiven Drogenhändler der Anstrengung wieder
Zehnmalten von Studenten aus der Peking-Universität zum Platz des
Himmelsfreundes marschiert, um für mehr Demokratie zu demonstrieren.

4]DE-MQN-[3]STR1,-R6N.

005
Sie erhielten bei ihrem Zug durch die Strassen Zufall von Pekinger Bürgern.


006
Trotz scharfen Warnungen der chinesischen Staats- und Parteiführung haben die
Studenten in Peking am Donnerstag eine weitere Massenkundgebung abgehalten.

2NT PAR-[1]TA1-F7-R29 #1] [2]HA,B6N DI; 5TU-[1]D3-N6N #2] IN

007
Die Disziplin der Studenten und das massvolle Verhalten der Polizei- und
Militärbehörden sorgte dafür, dass der über 30 Kilometer lange Marsch, an dem
nach rund 50'000 Hochschülern auch einige zehntausend Sympathisanten
teilnahmen, friedlich verlief.


F3,R-[3]L1,F.

008
Dialogsignal

009
Nachdem die von beiden Seiten befürchtete Konfrontation vermieden worden war,
signalisierte die Regierung bedingte Verständigungsbereitschaft.

ZIG-NA,1-[1]Z1:-R6-T6 D1; RE-[1]G1,-R29 #2] B6,2]D9-R6
F3,R-[1]T51N-D1-298-B6,4]RA1T-SAPT.

010
Ein Sprecher des Staatsrats, den das Fernsehen in den Abendnachrichten zitierte,
bot den Studenten einen Dialog an.


181
Nach eigenen Angaben sollen die Soldaten bis zu vier Millionen an der Macht bleiben.


182
Johannesburg.

183
Bß neuen Unruhen in verschiedenen südafrikanischen Städten sind drei Schwarz
ums Leben gekommen.

2MS [1]LE,B6N Ga,2]KQ,M6N.

184
Bei Protesten gegen das Verkehrchaos in Karatschi sind neun Menschen getötet worden.


185
Über hundert wurden verhaftet.


186
Neues Bulletin ab 10 Uhr 45.


Appendix D

SVOX Synthetic Speech Demonstration

This appendix presents a German text as it can be entered into the SVOX TTS system in order to produce synthetic speech. The corresponding speech signals are available via a world-wide web (WWW) page. Currently, all sentences must be terminated by ’.’, ’?’ or ’!’. Empty sentences, that is, individual full stops, produce pauses of 500 msec duration. As is shown in the weather forecast text (the second section of the text below), upper- and lowercase letters are not distinguished, and ’ä’, ’ö’, and ’ü’ can be replaced by ’ae’, ’oe’, and ’ue’.

Nordwind und Sonne.

musste der Nordwind zugeben, dass die Sonne von ihnen beiden der Stärkere war. 

....

erster wetterbericht von 5 Uhr 30.

allgemeine lage.
mit nordwestlichen winden wird heute maessig feuchte luft gegen unser land gefehrnt und an den alpen gestaut.
prognosen fuer heute donnerstag.
alpenmondseite, wallis und graubuenden, im flachland wechselnd bewoelkt mit aufhellungen. am alpennordhang starker bewoelkt.
och einzelne schauer. temperaturen am nachmittag um 22 grad.
mullgradgrenze gegen 3000 meter hoehe. in den bergen und zum teil auch im flachland maessiger nordwestwind. alpensuedseite, ziemlich sonnig. den alpen entlang zum teil bewoelkt.
tageshoehsttemperaturen um 27 grad.

wetteraussichten bis nachsten montag abend.
am freitag und samstag, im osten weiter abklingende schauerneigung und von westen her zunehmend sonnig. im westen ziemlich. im sueden sogar vorwiegend sonnig. am sonntag und montag: schonen und sommerlich warm. im norden einzelne abendgewitter.

....

Schneewittchen.

Es war einmal mitten im Winter, und die Schneeflocken fielen wie Federn vom Himmel herab, da sass eine Königin an einem Fenster, das einen Rahmen von schwarzen Ebenholz hatte, und nähte. Und wie sie so nähte und nach dem Schnee aufblickte, stach sie sich mit der Nadel in den Finger, und es fielen drei Tropfen Blut in den Schnee. Und weil das Rote im weissen Schnee so schön aussah, dachte sie bei sich: Hätt ich ein Kind so weiss wie Schnee, so rot wie Blut und so schwarzhaarig wie Ebenholz und wurde darum Schneewittchen genannt. Und wie das Kind geboren war, starb die Königin.

....

Über ein Jahr nahm sich der König eine andere Gemahlin. Es war eine schöne Frau, aber sie war stolz und übermütig und konnte nicht leiden, dass sie an Schönheit von jemand sollte übertroffen werden. Sie hatte einen wunderbaren Spiegel, wenn sie vor den trat und sich darin beschaute, sprach sie: "Spiegel, Spiegel an der Wand, wer ist die Schönste im ganzen Land?!", so antwortete der Spiegel: "Frau Königin, Ihr seid die Schönste im Land." Da war sie zufrieden, denn sie wusste, dass der Spiegel die Wahrheit sagte.

Schneewittchen aber wuchs heran und wurde immer schöner. Und als es sieben Jahre alt war, war es so schön wie der klare Tag und schöner als die Königin selbst. Als diese einmal ihren Spiegel fragte: "Spiegel, Spiegel an der Wand, wer ist die Schönste im ganzen Land?!", so antwortete er: "Frau König, Ihr seid die Schönste hier, aber Schneewittchen ist tausendmal schöner als Ihr."

Bibliography


H. Fujisaki. Dynamic characteristics of voice fundamental frequency in speech and singing. Acoustical analysis and


