Multifunktionale und multilinguale Unit-Selection-Sprachsynthese

Designprinzipien für Architektur und Sprachbausteine

Inauguraldissertation
zur Erlangung der Doktorwürde
der Philosophischen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität
zu Bonn

vorgelegt von
Stefan Breuer
aus Köln

Bonn, 2009

Gedruckt mit der Genehmigung der Philosophischen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Zusammensetzung der Prüfungskommission:

Prof. Dr. Caja Thimm
(Vorsitzende)

Prof. Dr. Wolfgang Hess
(Betreuer und Gutachter)

apl. Prof. Dr. Bernd Möbius
(Gutachter)

PD Dr. Ulrich Schade
(weiteres prüfungsberechtigtes Mitglied)

Tag der mündlichen Prüfung: 9. Mai 2008

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn unter
http://hss.ulb.uni-bonn.de/diss_online elektronisch publiziert.

Danksagung

Unlängst schrieb Papst Benedikt XVI. über eine neue Buchveröffentlichung, zu diesem Werk sei er „lange innerlich unterwegs gewesen“. Diese Aussage charakterisiert auch die Entstehung dieser Dissertation. Dass ich sie geschrieben habe verdanke ich vielen Leuten. Zu nennen ist selbstverständlich Prof. Wolfgang Hess, der mir die Gelegenheit gegeben hat, in seiner Gruppe zu arbeiten. Dank gebührt aber auch

- Dr. Thomas Portele, der mich erstmals in das Team des IKP aufgenommen hat,
- Dr. Petra Wagner für ein stets offenes Ohr für wissenschaftliche Fragen und Gejammer,
- Prof. Bernd Möbius für die Ermutigung, meine Arbeit schließlich auf Papier zu bringen,
- Dr. Karl-Heinz Stöber für BOSS und die Grundsteine der Multiphon-Entwicklung,
- der Firma klickTel GmbH für die finanzielle Unterstützung,
- den klickTel-Hilfskräften Meike, Katja, Nadja, Amanda, Christian, Harald und Boris,
- Daniela und Lioba für ihre Stimmen,
- den Hilfskräften in der BOSS-Entwicklung, Míchal, Philip und Denis,
- den Magistern und Diplom-Informatikern Robert, Hannes und Tim,
- den Hilfskräften Igor, Sebastian, Anne, Thomas und Till, die mir in der System- und Web-Administration den Rücken freigehalten haben,
- meiner Mutter für ihre fortwährende Unterstützung während der gesamten Qualifikationsphase
- und allen, die ihn verdienen und die ich vor Müdigkeit vergessen habe zu erwähnen.

Der größte Dank aber gebührt meiner Frau, Dr. Julia Abresch, die mich in der knappen Zeit der Niederschrift so um- und versorgt hat, als sei ich mit unserem Nachwuchs schwanger und nicht sie. Dir, Julia, widme ich diese Dissertation.

Inhaltsverzeichnis

1. Einleitung **1**

2. Unit-Selection-Sprachsynthese **5**
2.1. Die Anfänge: *Non-Uniform Unit Selection* für das Japanische 8
2.2. Der Durchbruch: Fokus auf die Variabilität des Segments 13
2.3. Die Verbmobil-Synthese 18
2.3.1. Das Einheitenauswahlverfahren der Verbmobil-Synthese 19
2.4. Weitere Systeme und Methoden 25
2.5. Forschungsschwerpunkte der Unit-Selection-Synthese 26
2.5.1. Auswahlalgorithmen und prosodische Manipulation 26
2.5.2. Abstandsmaße, Signalrepräsentation und Kostengewichte . . . 31
2.5.3. Korpusdefinition, Einheiten und Konkatenation 32

3. Neudefinition von Syntheseeinheiten **37**
3.1. Motivation 37
3.2. Phoxsy: Multiphon-Einheiten für die Einheitenauswahl aus großen Korpora 38
3.2.1. Beispiele für zusammengesetzte Lautfolgen 42
3.2.2. Entwicklung und Spezifikation 55
3.2.3. Kontextklassen 58
3.3. Evaluation durch Präferenztests 59
3.3.1. Auswahl der Stimulusvorlagen 60
3.3.2. Generierung der Stimuli 61
3.3.3. Analyse der Stimuli 62
3.3.4. Ergebnisse 63
3.4. Diskussion 64

4. Bonn Open Synthesis System **67**
4.1. Korpuserstellung und -repräsentation 67
4.1.1. *BOSS Label File* 67
4.1.2. BOSS-Inventar-XML 68
4.1.3. Relationale Datenbankstruktur 70

4.2. Synthesearchitektur . . . 71
4.2.1. XML als Repräsentationssprache . . . 72
4.2.2. Aufbau des BOSS-Servers . . . 74
4.2.3. Transkription . . . 75
4.2.4. Unit Selection . . . 76
4.2.5. Signalloader . . . 78
4.3. Fazit . . . 79

5. Ausbau der BOSS-Architektur: Multilingualität und Multifunktionalität 81
5.1. Konfiguration und Kommunikation . . . 82
5.1.1. Das Konfigurationskonzept . . . 82
5.1.2. Erweiterung des Netzwerkprotokolls . . . 85
5.1.3. Implementierung einer Dateischnittstelle . . . 87
5.2. Dynamisierung der Modulstruktur . . . 87
5.2.1. Funktionalität des deutschen Dauermoduls . . . 88
5.2.2. Überlegungen zur strukturellen Veränderung und Implementierung . . . 90
5.2.3. Systematisierung . . . 93
5.2.4. Empfehlungen für die Strukturierung von BOSS-Modulen . . . 95
5.2.5. Flexibilisierung der Moduleinbindung . . . 100
5.2.6. Struktur der Manipulationsmodule . . . 104
5.3. Unit Selection . . . 109
5.3.1. Einbindung der Kontextklassen . . . 109
5.3.2. Konfigurierbare Vorauswahl von Einheiten . . . 110
5.3.3. Kostenfunktionen . . . 112
5.3.4. Verarbeitung von Halbphon-Einheiten . . . 114
5.3.4.1. Einbettung in die XML-Repräsentation . . . 115
5.3.4.2. Erweiterung der Korpusaufbereitung und der Datenbank . . . 115
5.3.4.3. Veränderungen an den Laufzeitkomponenten . . . 116
5.3.4.4. Überlegungen zur Erweiterung auf Diphon-Synthese . 117
5.4. Prosodische Symbolverarbeitung . . . 120
5.4.1. Behandlung von Phrasengrenzen . . . 120
5.4.2. Anpassung der Pausenstruktur . . . 122
5.5. Weitere Entwicklungen und Ausblick . . . 124

6. Anpassung an eine Telefonauskunftsanwendung 129
6.1. Das Sprachkorpus . . . 130
6.1.1. Konzeption und Erstellung der Lesevorlagen . . . 130
6.1.2. Aufnahme und Annotation . . . 133

6.1.3. Zahlenprosodie . . . 135
6.1.4. Pruning . . . 136
6.1.5. Quantitative Analyse . . . 136
6.2. Aufbau eines Aussprachewörterbuchs . . . 138
6.2.1. Aussprache von Eigennamen . . . 138
6.2.2. Konventionen zur Transkription . . . 140
6.2.2.1. Wortakzent . . . 141
6.2.2.2. Vokalqualität und -quantität . . . 141
6.2.2.3. Behandlung von Silbenreduktion . . . 142
6.2.2.4. Fremdsprachliche Laute und Eigennamen . . . 142
6.2.3. Transkription der Namensdatenbank . . . 144
6.2.4. Bonn Machine-Readable Pronunciation Dictionary . . . 146
6.2.5. Erstellung des Gesamtlexikons für die Laufzeit-Synthese . . . 146
6.3. Laufzeitkomponenten der Synthese . . . 148
6.3.1. Textnormalisierung . . . 149
6.3.1.1. Eingabeformat . . . 149
6.3.1.2. Abkürzungsauflösung . . . 151
6.3.1.3. Vorverarbeitung von Zahlen . . . 153
6.3.2. Graphem-Phonem-Konvertierung . . . 153
6.3.2.1. Verfahren der automatischen Transkription . . . 154
6.3.2.2. Automatische Transkription von Eigennamen . . . 156
6.3.2.3. Auswahl eines Verfahrens . . . 157
6.3.2.4. Graphem-Phonem-Konvertierung für BOSS . . . 158
6.3.3. Vorauswahl von Einheiten . . . 163
6.3.4. Kostenfunktionen . . . 164
6.4. Evaluation der Synthese . . . 165

7. Zusammenfassung und Ausblick **169**

Abbildungsverzeichnis **176**

Tabellenverzeichnis **178**

Literaturverzeichnis **195**

A. BOSS-SAMPA/phoxsy **197**
A.1. Zusammensetzung der Einheiten für das Deutsche . . . 197
A.1.1. Vokale (Monophthonge) . . . 197
A.1.2. Nasalierte Vokale . . . 198
A.1.3. Diphthonge . . . 198
A.1.4. Konsonanten im Anlaut . . . 199

A.1.5. Approximant/Liquid + Schwa + Konsonant 199
A.1.6. Nasale 200
A.1.7. Frikative 200
A.1.8. Plosive 200
A.1.9. Affrikaten 201
A.2. Inventardefinition und Kontextklassen 202

B. BOSS-Konfigurationsoptionen 211
B.1. Auswirkung der Schaltereinstellungen 211
B.2. Auszug aus einer Konfigurationsdatei 211

C. XML-Schemata 213
C.1. Inventar-XML-Schema 213
C.2. Server-XML-Schema 219

D. Erweiterte SQL-Schemadefinition für BOSS-Datenbanken 225

E. Auskunft 235
E.1. Korpusbestandteile 235
E.2. Abdeckungsstatistiken zum Korpus 239
E.3. Abdeckungsstatistiken zum Eigennamenlexikon 239

1. Einleitung

„If you come to a fork in the road, take it." *Yogi Berra*

Sprachsynthese kann in den verschiedensten Bereichen eingesetzt werden und muss dabei divergierende Anforderungen erfüllen können. Die Bandbreite der Anwendungen reicht vom Vorlesen von Nachrichten über die sprachliche Wiedergabe von Bildschirminhalten und Menüstrukturen für Blinde bis zur Ausgabe von simulierter Spontansprache in Dialogsystemen. Dabei kommen verschiedene Methoden der Vorverarbeitung und Sprachsynthese zum Einsatz. Sollen mehrere solcher Anwendungsfelder und Verfahren von derselben Software abgedeckt werden und diese auch in der Lage sein, eine Ausgabe in unterschiedlichen Sprachen zu produzieren, muss das betreffende System möglichst flexibel gestaltbar sein und eine breite Palette an grundlegenden Algorithmen inkorporieren. Diese Arbeit beschäftigt sich mit einer solchen Synthesearchitektur für viele Sprachen und Anwendungen. Aus diesem Grunde wird das schon den Titel dominierende Morphem *multi* dem Leser in den folgenden Kapiteln noch des öfteren begegnen.

Die Grundlage für die eigenen Arbeiten zum Aufbau einer multilingualen und multifunktionalen Synthese bildet das von Karlheinz Stöber begonnene *Bonn Open Synthesis System*, BOSS (Stöber, 2002). Das Ziel dieser Entwicklung war, eine quelloffene Basis für die Forschung und Entwicklung von Syntheseanwendungen nach dem Prinzip der konkatenativen Synthese von großen Korpora zu schaffen. Dieses auch *Unit Selection* genannte Verfahren der maschinellen Erzeugung von Sprache bedient sich der Rekombination einer Vielzahl von zuvor aufgenommenen, natürlichen „Sprachschnipseln". Einer der Schwerpunkte dieser Arbeit ist daher die Frage, wie diese Bausteine beschaffen sein müssen, um gleichzeitig die Natürlichkeit der Synthese erhöhen und die Erstellung der benötigten Sprachdatenbanken vereinfachen zu können. Dass BOSS aber auch ein Rahmen für andere Verfahren, wie z. B. die artikulatorische Synthese sein kann (Birkholz et al., 2007), ist u. a. ein Resultat der Entwicklungen, die den zweiten Fokus dieser Arbeit darstellen: Die Flexibilisierung der BOSS-Architektur und die Erweiterung um Komponenten und Funktionen, die die Entwicklung von BOSS in Richtung einer allgemeinen, sprachabstrahierenden und funktional heterogenen Synthese vorangetrieben haben und es nun ermöglichen,

das System gleichzeitig an verschiedene Sprachen und Anwendungen anzupassen. Dabei spielten theoretische Überlegungen zur Modularisierung und Wiederverwendbarkeit von Synthesekomponenten eine wichtige Rolle. Ein Teil der notwendigen Veränderungen wurde im Rahmen der Erstellung eines Systems zur Wiedergabe von Namen, Adressen und Rufnummern in einem telefonischen Auskunftssystem vorgenommen. Diese Art der Anwendung stellt besondere Anforderungen an die Struktur der Synthese, insbesondere aber an die Beschaffenheit der verwendeten linguistischen Ressourcen. Die inhaltliche Planung des Aussprachewörterbuchs und der Sprachdatenbank, deren Erstellung sowie die Erweiterung der Schnittstelle zwischen Orthographie und phonetischer Repräsentation für die Auskunftssynthese bilden daher den dritten und letzten großen Schwerpunkt der Arbeit. Damit ist ein breites Spektrum der Aspekte multilingualer und multifunktionaler Synthese abgedeckt.

Der Aufbau der Arbeit ist der folgende: In Kapitel 2 werden zunächst die Grundlagen für das Prinzip der konkatenativen Synthese erläutert. Es folgt ein historischer Abriss darüber, wie dieses Prinzip zum Verfahren der Unit-Selection-Synthese weiterentwickelt wurde. Besondere Beachtung findet in dieser Beschreibung das Verbmobil-System (Stöber et al., 2000), dessen grundlegende Methoden auch die Basis für die Einheitenauswahl von BOSS sind. Im Anschluss werden weitere Systeme präsentiert, die zuvor oder parallel entwickelt wurden. Im letzten Teil des Kapitels werden die wichtigsten Forschungsfelder, Algorithmen und Herausforderungen der Einheitenauswahl-Synthese dargestellt.

Kapitel 3 vertieft die Diskussion zur Problematik der Einheitendefinition und stellt ein eigenes Konzept sowie eine daraus abgeleitete Spezifikation von Synthesebausteinen für das Deutsche vor. Die Funktion dieser Bausteine wird zunächst theoretisch diskutiert und an Signaldarstellungen erläutert. Der letzte Abschnitt schildert die Vorgehensweise zu einer globalen Evaluation der Einheitenspezifikation.

Als Grundlage für das Verständnis der Erweiterungen wird in Kapitel 4 die Ausgangsarchitektur von BOSS beschrieben. Dabei wird auch die Bedeutung von XML-Sprachen als Kommunikations- und Repräsentationsformat für die Sprachsynthese diskutiert.

Den Ausbau des BOSS-Systems zu einer multilingualen und multifunktionalen Architektur beschreibt Kapitel 5. Die Darstellung umfasst die Erweiterung und Entwicklung einer Vielzahl von Komponenten aus allen Teilaufgaben der Synthesesoftware. Zusätzlich werden formalisierte Empfehlungen für den Aufbau zukünftiger Module gegeben.

Kapitel 6 beschäftigt sich insbesondere mit den Ressourcen, die für die Erstellung der Telefonauskunftsanwendung geschaffen wurden und den Methoden, um die Beziehung zwischen der Annotation dieser Ressourcen und der Adressdaten-

bank durch Textnormalisierung und automatische Transkription herzustellen. Dazu werden zunächst Konzeption und Inhalt des Sprachkorpus beschrieben. Darauf folgt die Schilderung der Konventionen und der Strategie zum Aufbau des Aussprachewörterbuchs. Die Laufzeitkomponenten zur Symbolverarbeitung stehen im Mittelpunkt der zweiten Hälfte des Kapitels. Im letzten Abschnitt dieses Teils wird eine Evaluation des Systems präsentiert.

Abschließend wird in Kapitel 7 der Inhalt der Arbeit zusammenfassend dargestellt und ein Ausblick auf zukünftige Forschungs- und Entwicklungsaufgaben gegeben.

2. Unit-Selection-Sprachsynthese

„Der Schwache zweifelt vor der Entscheidung, der Starke danach.“ *Karl Kraus*

Zahlreiche Einführungen bieten einen Überblick über die Geschichte künstlich erzeugter Sprache und die unterschiedlichen Verfahren, die in der maschinellen Sprachsynthese verwendet werden (bspw. Hess, 2002; Dutoit, 1997; Klatt, 1987), Taylor, 2007). In diesem Kapitel soll daher nur das derzeit in der Forschung und industriellen Anwendung vorherrschende Synthese-Paradigma betrachtet werden, nämlich das der Einheitenauswahl aus großen Korpora oder kurz, der *Unit Selection*. Dieser noch relativ junge Ansatz gehört zu den datenbasierten oder konkatenativen Verfahren, die synthetische Sprache durch die Verkettung von zuvor aufgenommenen, natürlichen Sprachsignalen erzeugen. Einige Grundkonzepte der Sprachsynthese im Allgemeinen sowie der konkatenativen im Besonderen sollen jedoch vorausgeschickt werden.

Die Standardanwendung für die maschinelle Sprachsynthese ist die Erzeugung von gesprochener Sprache aus geschriebenem, digital repräsentiertem Text, genannt *text-to-speech* (TTS). Damit ein solcher Vorleseautomat diese Aufgabe leisten kann, sind viele Einzelschritte notwendig (s. Abbildung 2.0.1). Da der Text fast ausschließlich Informationen zur lautlichen Struktur enthält, die durch das Missverhältnis zwischen Orthographie und Aussprache je nach Sprache auch noch mehr oder weniger inkonsistent sind, müssen alle prosodischen Parameter, also die Satzmelodie oder Intonation, ausgedrückt durch den akustischen Parameter Grundfrequenz (F0), die Dauern der Einzellaute und ggf. auch deren Lautstärke geschätzt und vorhergesagt werden. Die frühen TTS-Systeme verfügten über Regelwerke, die den Bestandteilen der Äußerungen diese Werte zuwiesen. Heutzutage dominieren statistische Verfahren, mit denen die Systeme anhand von großen prosodisch und phonetisch etikettierten (annotierten) Datenbanken lernen, bestimmten Satz- und Lautstrukturen typische F0- und Dauerwerte zuzuweisen (bzw. diese zu prädizieren). Auch die tatsächlichen Lautbestandteile (Phone oder Segmente genannt) der zu synthetisierenden Äußerung werden oft mit Hilfe solcher Klassifikationsalgorithmen ermittelt, um die Orthographie in eine lautschriftliche Darstellung (phonetische Transkription) zu übersetzen. Diese ersten Schritte der TTS-Synthese werden zusammenfassend

auch Symbolverarbeitung genannt. In der sogenannten *content-to-speech*-Synthese (CTS) ist die Grundlage nicht ein reiner orthographischer Text, sondern eine partielle oder vollständige phonetische und prosodische Spezifikation der Äußerung, die von Hand oder einem anderen Programm erzeugt wird. Dadurch kann die Symbolverarbeitung bei CTS zumindest teilweise entfallen.

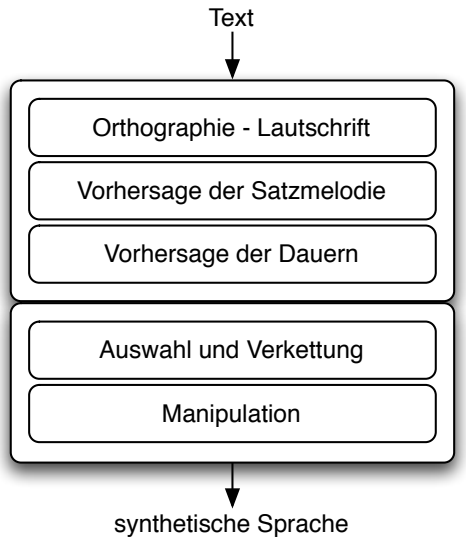

Abbildung 2.0.1.: Stark vereinfachtes Schema eines Vorleseautomaten nach dem Prinzip der konkatenativen Synthese.

Da die Zahl der Wörter einer Sprache prinzipiell unbeschränkt, wenn auch, wie Stöber (2002) argumentiert, aus zeitlichen Gründen praktisch endlich ist, können nicht einfach alle möglichen Wörter aufgenommen und wieder abgespielt werden, selbst wenn die erhebliche Variabilität in der Aussprache und Prosodie der Wörter außer acht gelassen wird. Das Lautinventar einer Sprache ist jedoch beschränkt, so dass zunächst Versuche unternommen wurden, durch Rekombination einzelner aufgenommener Laute neue Äußerungen zu erzeugen. Da die Artikulation aufeinanderfolgender Sprachlaute eng miteinander verzahnt ist (Koartikulation, Menzerath und de Lacerda, 1933) und die akustischen Übergänge zwischen den Lauten daher entscheidende Informationen zur Erkennung liefern, scheiterte dieser Ansatz. Erst mit der Einführung von sogenannten Diphonen (Küpfmüller und Warns, 1956; Peterson et al., 1958), das sind Einheiten, die aus dem Übergang zweier Laute bestehen, konnte die Entwicklung der datenbasierten Synthese fortgesetzt werden (s. Abbildung 2.0.2). In der Folge wurden weitere Einheitentypen, wie bspw. Halbsilben definiert und es entstanden Sprachdatenbanken (auch Korpora, Inventare oder ,,Stimmen" genannt), in denen verschiedene Typen koexistierten (Dettweiler und Hess, 1985; Portele, 1996). Für jeden unterschiedlichen Laut oder Lautübergang existierte dabei

genau eine Einheit im Korpus. Diese Art der konkatenativen Synthese wird häufig als Diphonsynthese bezeichnet, auch wenn zusätzlich andere Einheitentypen verwendet werden. Hier soll im Folgenden von der klassischen oder traditionellen konkatenativen Synthese die Rede sein, wenn es um die Abgrenzung zur Unit-Selection-Synthese geht.

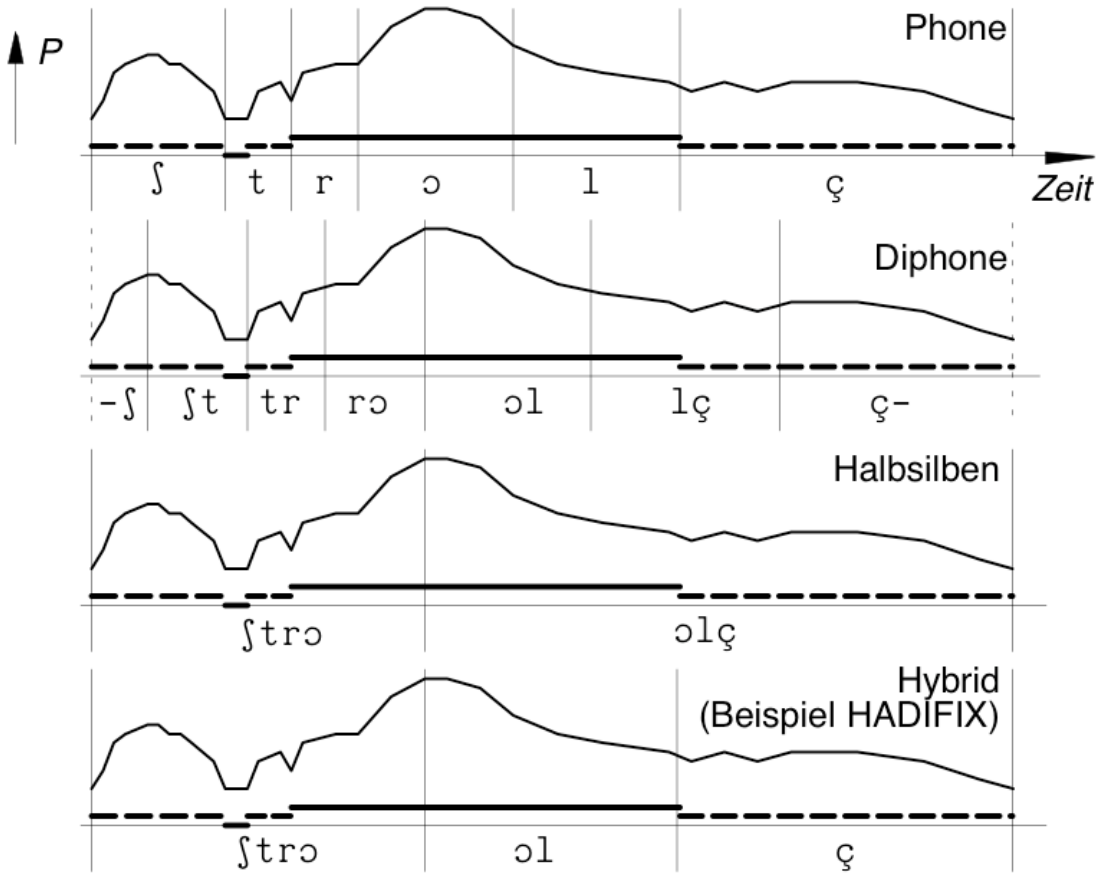

Abbildung 2.0.2.: Synthese des Wortes „Strolch" [ʃtrɔlç] mit Hilfe verschiedener phonetischer Einheiten. (P) Beliebiger Signalparameter (hier gezeichnet: Pegel und Stimmhaftigkeit) – Hess (2002)

In der zweiten Stufe der klassischen konkatenativen TTS-Synthese wird für jeden Laut in der lautschriftlichen Zielvorgabe die entsprechende Einheit aus dem Korpus extrahiert. Dies wird durch eine phonetische Annotation (Labelling) der Korpora ermöglicht, meist bestehend aus einfachen Textdateien, die für jede Einheit eine Beschreibung und Zeitmarken für Beginn und Ende im Signal (der Audiodatei) enthalten. Die einzelnen Einheiten werden in der gewünschten Reihenfolge zu der zu synthetisierenden Äußerung verkettet (konkateniert). Um die Äußerung tatsächlich verständlich zu machen, werden die Dauern und die F0-Werte der Kette noch durch

Manipulation des Signals der prädizierten Vorgabe angepasst.

Im weiteren Verlauf des Kapitels werden zunächst die historischen Ursprünge der um Unit Selection erweiterten konkatenativen Synthese behandelt und deren Entwicklung bis hin zu den aktuellen Sprachsyntheseverfahren des Instituts für Kommunikationswissenschaften verfolgt, bevor parallele Ansätze der Einheitenauswahl in der Industrie und akademischen Entwicklung und schließlich die Schwerpunkte der Unit-Selection-Forschung besprochen werden.

2.1. Die Anfänge: Non-Uniform Unit Selection für das Japanische

Verschiedene Quellen sind sich einig (Black und Lenzo, 2008; Möbius, 2000), dass die Ursprünge der Unit Selection in den japanischen ATR Laboratories Ende der 1980er Jahre zu suchen sind. Dort entwickelte Sagisaka 1988 ein Syntheseverfahren für das Japanische, das unterschiedlich lange Bausteine, sogenannte *non-uniform units* zu einer Äußerung zusammensetzte. Daran war weniger der Umstand revolutionär, dass die Einheiten nicht uniform waren, da auch andere konkatenative Systeme zu dieser Zeit schon verschiedene Einheitentypen kombinierten, beispielsweise durch Affixe ergänzte Halbsilben (Dettweiler und Hess, 1985), sondern dass für jeden zu generierenden Äußerungsabschnitt mehrere alternative Bausteine zur Wahl standen, die zur Laufzeit nach phonetischen Kriterien ausgewählt wurden. Im Kern ist es diese Eigenschaft, die auch heute noch ein Unit-Selection-System ausmacht. Die Basis für diese alternativen Bausteine waren bei Sagisaka nach dem Verfahren der linearen Prädiktion (LPC, Atal und Hanauer, 1971) kodierte Einzelphone, die jedoch in zusammenhängenden Ketten ausgewählt und mit anderen Ketten unterschiedlicher Länge zusammengefügt werden konnten. Die Auswahl der einzelnen Ketten verlief nach folgendem Muster:

Für jede Phrase einer Zieläußerung werden alle möglichen Zerlegungen der segmentalen Struktur ermittelt. Dann werden aus einer *synthesis unit entry dictionary* genannten Baumstruktur alle im Korpus enthaltenen Lautfolgen extrahiert, die den Zerlegungen entsprechen (s. Abbildung 2.1.1). Von diesen Folgen, nachfolgend Einheiten genannt, werden nun die ausgewählt, die sich am besten in den Kontext einfügen. Kriterien dafür sind die Anzahl der überlappenden Laute in den hypothetischen Nachbareinheiten, die möglichst hoch sein sollte, sowie eine minimale spektrale Störung an den prospektiven Verkettungsstellen[1]. Nur die Einheiten, die

[1]Diese wird allerdings nicht für jede individuelle Konkatenationsstelle berechnet. Stattdessen wird ein Mittelwert verwendet („averaged spectral distortions", Sagisaka, 1988).

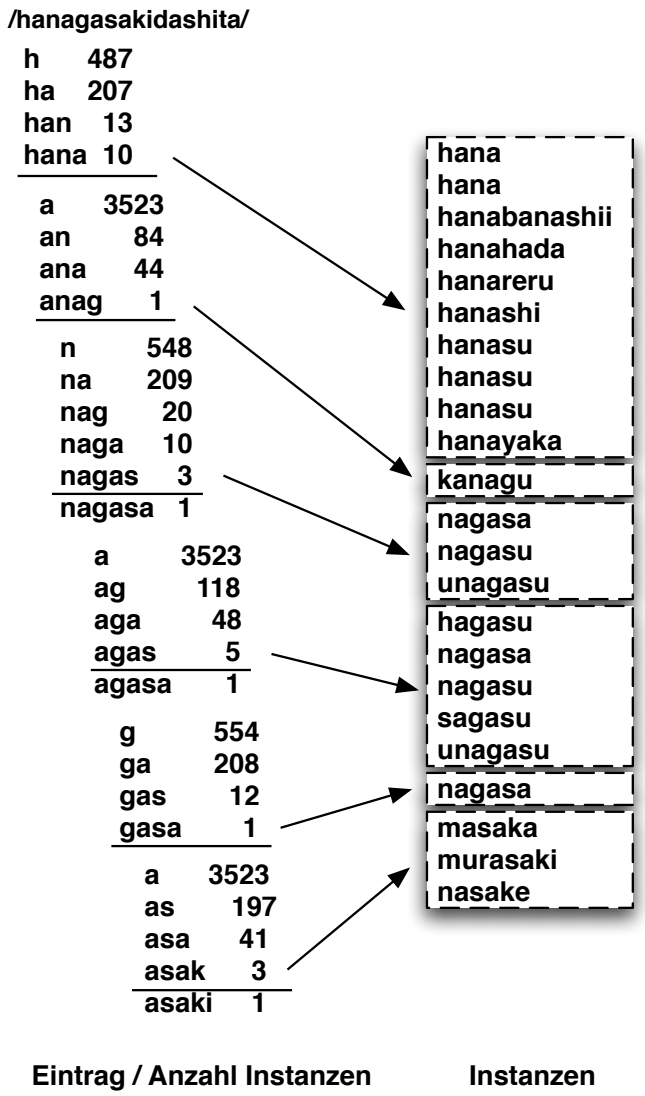

Abbildung 2.1.1.: Zerlegung der Eingabe /hanagasakidashita/ in alle möglichen Lautfolgen mit anschließender Auswahl passender nicht-uniformer Einheiten aus dem Korpus – modifizierte Darstellung nach Sagisaka (1988).

einen bestimmten Schwellwert überschreiten, werden als Kandidaten für die Verkettung in Betracht gezogen. Aus den verbleibenden Einheiten wird nun eine Abfolge gesucht, die die Anzahl der Konkatenationsstellen minimiert, möglichst geeignete Verkettungspunkte enthält und dabei Koartikulationseffekte erhält. Um dies zu erreichen werden Verkettungen an Vokal-Konsonant-Übergängen gegenüber solchen an Konsonant-Vokal- und Vokal-Vokal-Transitionen bevorzugt und lange Einheiten mit hohen Überlappungen zu den potenziellen Nachbarn besser bewertet. Ausgedrückt werden diese Präferenzen in Form von niedrigen *Kosten* für die als besonders geeig-

net betrachteten Einheiten und hohen Kosten für schlecht geeignete Einheiten. Am Ende wird die Kombination von Einheiten gewählt, die die Gesamtkosten minimiert. Diese wird, wie in der klassischen konkatenativen Synthese, nach den Vorgaben der Zieläußerung in Dauer und Frequenz manipuliert. Der Aufbau des Gesamtsystems wird in Abbildung 2.1.2 dargestellt.

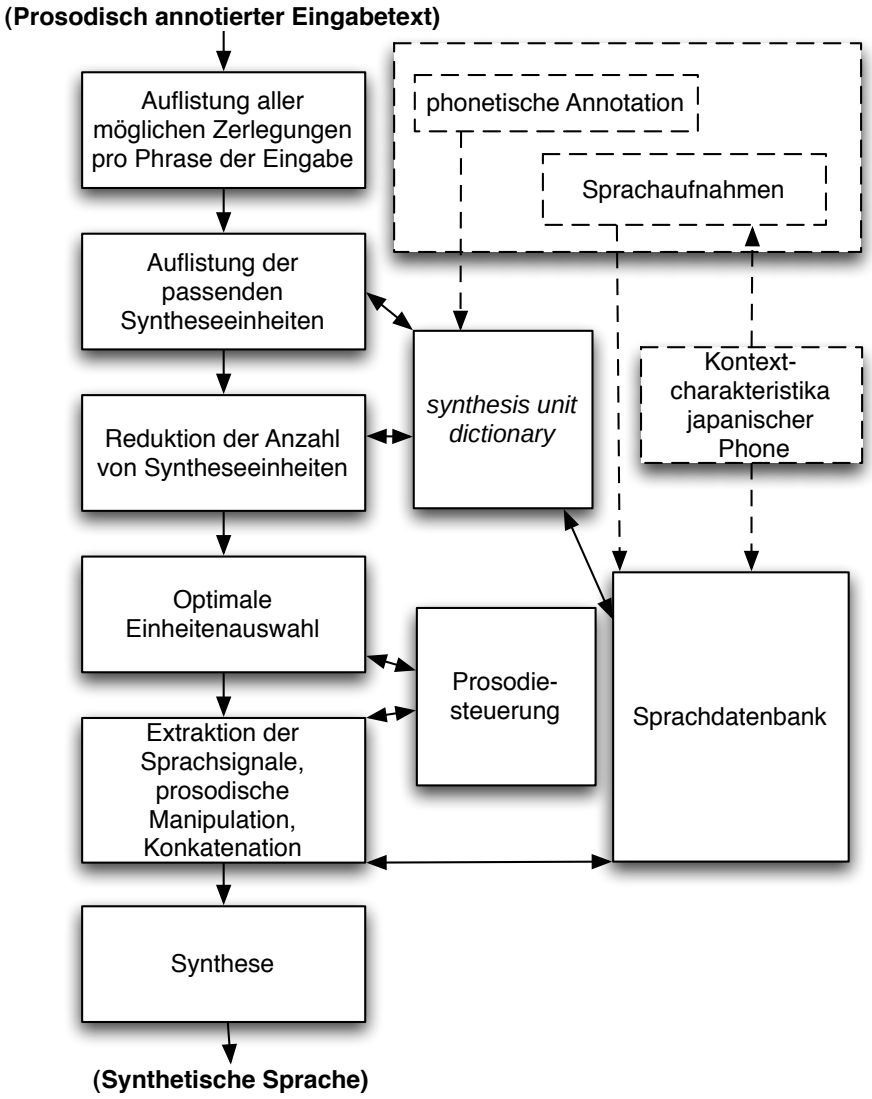

Abbildung 2.1.2.: Aufbau des ersten Systems nach dem Prinzip der Non-Uniform-Unit-Selection. Der geklammerte Text oben entspricht der Symbolverarbeitungskomponente, die Kästen links ersetzen die Verkettungs- und Syntheseanteile der traditionellen konkatenativen Synthese – modifiziert nach Sagisaka (1988).

Zwei Varianten der Auswahlmethode und Kostenfunktionen für Sagisakas System zur *Non-Uniform Unit Selection* werden von Takeda et al. (1990) und Sagisaka

(1989) beschrieben. Auch hier werden solche Kosten unterschieden, die sich einerseits auf die Kontinuität zwischen Einheiten und andererseits auf die kontextabhängige spektrale Struktur eines Auswahlkandidaten beziehen, um auch Koartikulationseffekte zu berücksichtigen, die sich über die gesamte Kandidateneinheit erstrecken. Bei der *single cost function* (SCF) wird im Unterschied zur Methode in (Sagisaka, 1988) keine Vorauswahl der Einheiten getroffen. Für die Zieläußerung wird jede mit der Datenbank realisierbare Verkettung, mit der diese Äußerung erzeugt werden kann, auf ihre Eignung überprüft. Dies geschieht, indem um jedes Kandidatensegment im Korpus ein Fenster von jeweils drei Phonen vor und nach der Einheit gelegt wird. Jedes reale Kontextphon wird nun mit dem entsprechenden Kontextphon der Zieläußerung verglichen und dabei nach nicht näher beschriebenen Kriterien aufgrund seiner Eignung als Lautkontext und als Konkatenationsstelle mit Kosten belegt. Dabei werden die Kosten der Kontextphone mit steigendem Abstand von der Einheit geringer gewichtet. Kontinuitäts- und Koartikulationskriterien werden aufaddiert und die Einheitenfolge mit den niedrigsten Kosten zur Synthese der Zieläußerung ausgewählt.

Die zweite Methode, *top down method selection* (TDH), nimmt zunächst eine Vorauswahl nach Kontinuitätskriterien vor. Dabei wird eine hypothetische (mutmaßlich nach den Kriterien optimale) Zerlegung der Zieläußerung vorgegeben. Dann wird abgefragt, ob diese Struktur mit den Lautfolgen in der Datenbank generiert werden kann. Geht dies nicht, wird die nächstbeste Hypothese untersucht, bis eine realisierbare Verkettung gefunden wird. Alle Kandidaten für diese Verkettung werden entsprechend der SCF, jedoch nur unter Berücksichtigung der Kontextkriterien, mit Kosten versehen und die beste Sequenz von Kandidaten wird ausgewählt.

In der Evaluation der beiden Algorithmen schnitt die TDH in den subjektiven Tests zur Verständlichkeit und Präferenz besser ab als die SCF (jeweils verglichen mit CV-Silben), obwohl die Einheitenlänge bei Letzterer im Mittel höher ausfiel. Auch waren bei der objektiven Evaluation im Schnitt die spektralen Unterschiede an den Konkatenationsstellen (gemessen in mittleren Cepstrum[2]-Distanzen) und die Kontextkosten für die Top-Down-Methode geringer. Offenbar wirkte sich diese messbar höhere Gewichtung der perzeptiv auffälligeren Diskontinuitäten positiv auf die Urteile der Probanden aus. Über einen weiteren Faktor kann gemutmaßt werden: Im Gegensatz zur SCF verfügt die TDH nämlich auch über wenigstens ein akustisch-prosodisch motiviertes Maß: den Abstand zwischen dem prädizierten Ziel-F0-Wert einer Einheit und dem der Kandidaten.

Sagisaka et al. stellten 1992 das ν-Talk-System vor, das die Weiterentwicklungen der Non-Uniform Unit Selection am ATR inkorporierte. Neu waren insbesondere die

[2]Bogert et al. (1963)

vorherige Definition der nicht-uniformen Einheiten durch Vorverarbeitung des Korpus (Iwahashi und Sagisaka, 1992), s. auch Abschnitt 2.5, sowie Kostenfunktionen (Abbildung 2.1.3), die stärker auf akustische Kriterien abzielten (Iwahashi et al., 1992).

Der integrierte Selektionsmechanismus ist eigentlich eine Kombination aus einer zusätzlichen Datenbankreduktion (Pruning) und einer zweistufigen Auswahl. Er beinhaltet jeweils zwei Kostenmaße zur Abschätzung der Diskontinuität und der kontextbedingten spektralen Unterschiede. Das Pruning findet nicht zur Laufzeit der Synthese statt, sondern wird zuvor *offline* am Korpus vorgenommen. Ziel des Prunings ist es, nicht-prototypische Varianten eines Segments aus der Datenbank zu entfernen. Damit sind solche Segmente gemeint, deren zeitnormalisierte Cepstral-Werte einen großen (euklidischen) Abstand vom Zentroiden aller Segmente desselben Kontextes aufweisen.

Online wird im ersten Schritt die verbleibende kontextbezogene Kostenfunktion berechnet (*contextual spectral difference*, kurz CSD). Dabei wird der Vokal einer Kandidateneinheit, wiederum anhand von Cepstral-Werten, mit einer im Korpus vorhandenen Instanz desselben Vokals im Zielkontext verglichen. Dies ist insbesondere dann wichtig, wenn der Kandidat einem anderen Lautkontext entspringt, aber in Bezug auf Kontinuitätskriterien Vorzüge gegenüber einer Einheit mit exaktem Zielkontext aufweist. Existiert der exakte benötigte Vergleichskontext nicht im Korpus, wird auf eine ähnliche Instanz zurückgegriffen.

Zu den CSD-Kosten werden im gleichen Schritt die sogenannten *Disconcatability*-Kosten addiert. Diese stellen ein Kontinuitätsmaß dar, das über den Mittelwert der spektralen Veränderung an der Grenze zweier Lauttypen die Eignung dieses Punktes als Verkettungsstelle angeben soll. Dabei werden starke Veränderungen als Kriterium für gute Konkatenierbarkeit angesehen, weil sie einen Indikator für unterschiedliche Lautklassen vor und nach der Verkettungsstelle darstellen. Die besten Kandidateneinheiten für die nach dieser Maßgabe optimalen Sequenz von Einheitentypen werden für den zweiten Auswahlschritt vorgemerkt. Kontinuitätskosten werden bei dieser Vorauswahl höher gewichtet als die Kontextkosten.

Der zweite Schritt wählt für die Sequenz mit den im ersten Schritt ermittelten Konkatenationspunkten die Folge von Kandidaten aus, deren jeweiliger spektraler Abstand zueinander in der Summe am geringsten ausfällt (*Discontinuity*-Kosten). Dabei wird im Unterschied zur Disconcatability-Messung der geringstmögliche Abstand als optimal betrachtet.

Auch wenn ein direkter Vergleich der verschiedenen Entwicklungsstufen der Auswahlalgorithmen bis hin zur ν-Talk-Variante nicht vorliegt, kann doch angenommen werden, dass die jeweils aktuellste Version Verbesserungen in der Synthesequalität

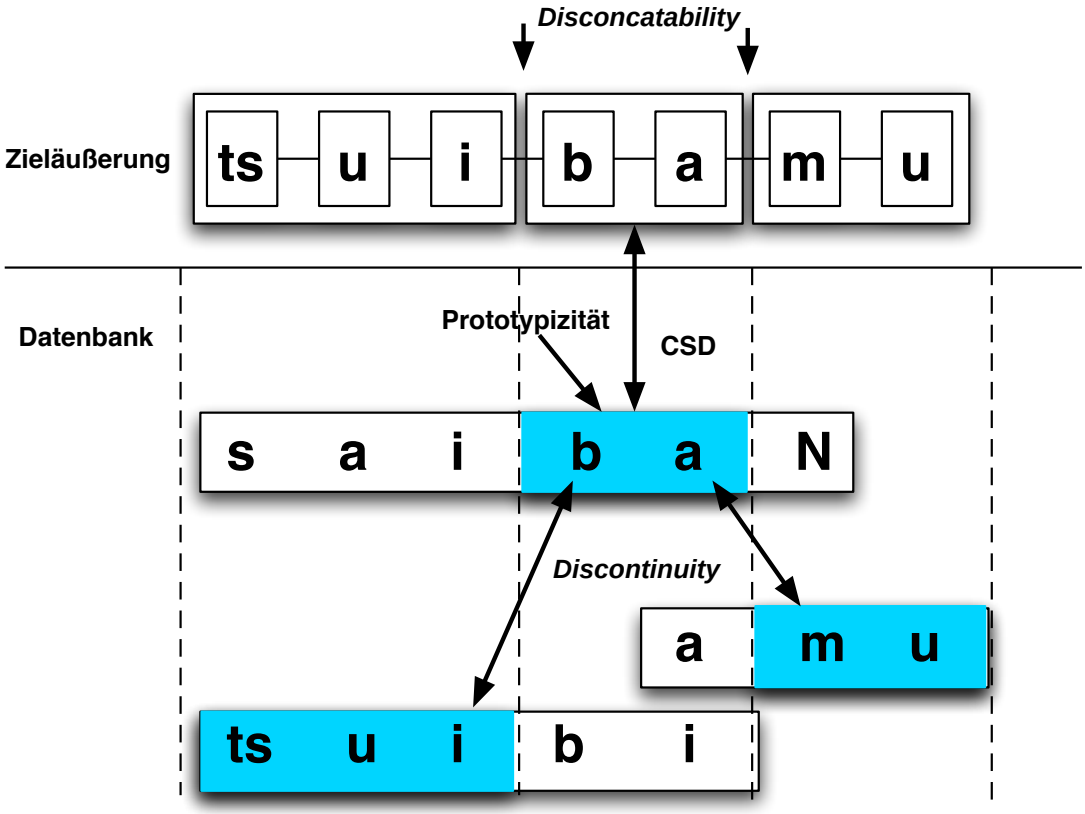

Abbildung 2.1.3.: Anwendung der Kostenterme des ν-Talk-Systems – modifizierte Darstellung nach Iwahashi et al. (1992).

brachte. Insbesondere die Ablösung der heuristischen phonetischen Kriterien durch akustische Abstandsmessungen muss als wichtiger Schritt in der Optimierung der Auswahlmechanismen angesehen werden.

Dennoch war die Non-Uniform Unit Selection von ν-Talk kein universell anwendbares Syntheseverfahren. Dies mussten auch Black und Campbell (1995) feststellen, als sie das System an das Englische zu adaptieren versuchten.

2.2. Der Durchbruch: Fokus auf die Variabilität des Segments

Rückblickend erscheint unmittelbar einleuchtend, dass die im vorangegangenen Abschnitt geschilderten Methoden nicht geeignet sind, die optimale Auswahl von Einheiten zu finden, wenn das Ziel die Ausgabe natürlich klingender und damit variantenreicher Sprache ist. So wurde jegliche nicht durch Koartikulation hervorgerufene

segmentale und prosodische Variabilität nicht nur in der Modellierung durch die Einheitenkosten vernachlässigt, sondern sogar durch Vorverarbeitungsstufen wie die Prototypisierung aktiv bekämpft. Iwahashi et al. (1992) schreiben zur Motivation dieser Maßnahme:

> ,,Speech signals show considerable variation, even in the same context, but irregular or uncharacteristic speech samples shouldn't be used."

Segmentale Reduktion und kontextbedingte Wechsel der Stimmqualität wurden damit in der Auswahl eher unterdrückt denn gefördert und es existierten in ν-Talk für diese Parameter, anders als für Dauer und Grundfrequenz, auch keine Mechanismen zur nachträglichen Signalmanipulation. Campbell und Black (Black und Campbell, 1995; Campbell und Black, 1996) erkannten, dass die vorhandenen Mechanismen für Sprachen, die komplexere prosodische Strukturen und satzphonetische Auswirkungen auf die Segmente aufweisen als das Japanische, nicht ausreichten. Daher erweiterten sie die Kostenfunktionen um prosodische Abstandsmaße und implementierten diese in der ebenfalls am ATR entwickelten, jedoch multilingual ausgelegten Syntheseplattform CHATR (Black und Taylor, 1994).

Bei ν-Talk und Vorgängern existierten Typen von Kosten, die sich mit den Eigenschaften der Einheit im Kontext befassen und Verkettungskosten, die sich auf die Kompatibilität der spektralen Eigenschaften um einen möglichen Konkatenationspunkt beziehen. Erstere messen die Unterschiede zwischen einer Kandidateneinheit und einem gewünschten Zielkontext, während letztere die Übergänge von einem Kandidaten zum nächsten in allen in Frage kommenden Zielsequenzen beurteilen (s. Abbildung 2.2.1).

Die prosodischen Erweiterungen in CHATR wurden von Black und Campbell (1995) für beide Typen vorgenommen. Begrifflich wurden damit aus dem reinen *spectral pattern difference* (Iwahashi und Sagisaka, 1992) die sogenannten *unit distortions*, während die *segment discontinuity* (ibid.) zu *continuity distortions* verallgemeinert wurde. Die assoziierten Kostenarten nennen Hunt und Black (1996) später *target costs* und *concatenation costs*. Im Folgenden sollen diese Typen Einheiten- und Transitionskosten genannt werden.

Einheiten sind in CHATR nach einem ähnlichen Verfahren wie bei ν-Talk vordefinierte, nicht-uniforme Folgen von Phonen — ,,typically a phone-sized segment" (Black und Campbell, 1995). Die verwendeten Eigenschaften für die Einheitenkosten sind der phonetische Kontext, die Dauer, die Intensität und die mittlere Grundfrequenz. In der Kostenberechnung werden diese in den Merkmalsvektoren gespeicherten Eigenschaften der Kandidateneinheiten im Korpus mit den Eigenschaften der Zieläußerung verglichen, die in den vorgeschalteten Komponenten der Synthese, wie der Transkription und den Modulen zur F0- und Dauerprädiktion geschätzt werden.

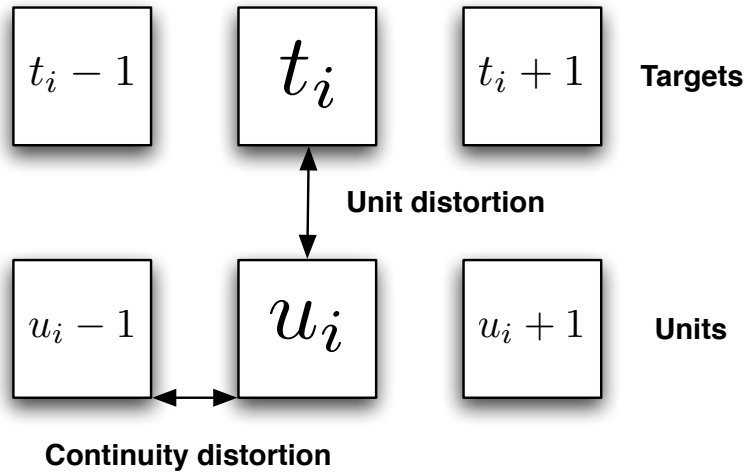

Abbildung 2.2.1.: Schematische Darstellung des Konzeptes der Einheiten- und Transitionskosten – nach Campbell und Black (1996).

Die Einbeziehung des F0-Wertes in den Vergleich zwischen Kandidateneinheit und Ziel war dabei auch gegenüber den vorherigen Entwicklungen am ATR eigentlich kein Novum, weil auch in den Arbeiten von Takeda et al. (1990) diese Kosten schon erfolgreich angewendet worden waren (TDH, s. Abschnitt 2.1). Allerdings fanden sie im später entwickelten ν-Talk-System offenbar keine Verwendung.

Wie bei ν-Talk wird die Zahl der Kandidaten für die Unit Selection durch eine Vorauswahl eingeschränkt. Dabei wird für jede Einheit der Zieläußerung nach Einheiten gesucht, die, wenn möglich, in Bezeichnung und unmittelbarem phonetischen Kontext (mindestens einseitig) identisch mit der Zieleinheit sind. Die Menge der so ermittelten Kandidaten wird dann noch einmal auf die nach den gewichteten Einheitenkosten 20 bis 50 besten Kandidaten reduziert.

Die Bestimmung der optimalen Einheit aus den so ermittelten Kandidaten für jede Zieleinheit — und damit der optimalen Folge von Einheiten zur Synthese der Zieläußerung — hängt auch von den Transitionskosten ab. Diese werden zum einen durch den spektralen Vergleich des idealen Konkatenationspunkts zweier aufeinanderfolgender Kandidaten anhand von mel-skalierten Cepstrum-Koeffizienten (*mel frequency cepstrum coefficients*, MFCC, Mermelstein, 1976) ermittelt. Zusätzlich wird aber auch der Originalkontext im Korpus, aus dem ein Kandidat stammt, nach den gleichen Kriterien wie bei den Einheitenkosten mit den hypothetischen Vorgängereinheiten und deren Korpuskontext verglichen. Dies wird innerhalb eines Fensters von zwei Einheiten um die untersuchte Konkatenationsstelle durchgeführt. Ähnlich wie bei den Einheitenkosten werden nur die 20 bis 50 Einheitenpaare mit den geringsten Transitionskosten für die Bestimmung der Bausteinfolge berücksichtigt.

Abbildung 2.2.2 (aufbauend auf Black und Campbell, 1995) illustriert diese zweite Komponente der Transitionskostenberechnung, die als Verschmelzung der *Discontinuity*-Funktion von Iwahashi et al. (1992) und einer erweiterten *single cost function* (SCF) nach Takeda et al. (1990) angesehen werden kann (vgl. Abschnitt 2.1). Dabei gibt es zwei wesentliche Unterschiede zur SCF: zum einen den schon erwähnten Vergleich akustisch-prosodischer Parameter, zum anderen, dass dieser zwischen Kandidaten und dem Kontext der Vorgängerkandidaten durchgeführt wird – also zwischen realen Signalen und nicht zwischen einem Signal und einer geschätzten Zielspezifikation.

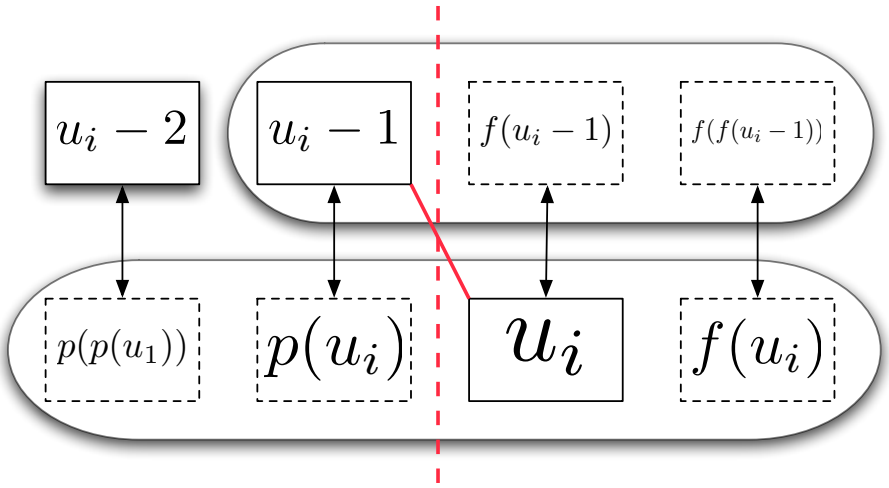

Abbildung 2.2.2.: Berechnung der prosodischen und phonetischen Unterschiede zwischen ausgewählten Kandidaten und ihrem jeweiligen Ursprungskontext in der Datenbank innerhalb der Transitionskostenfunktion. Abb. aufbauend auf Black und Campbell, 1995)

$p()$ ist eine Funktion zur Ermittlung des vorausgehenden Kontextes eines Kandidaten im Korpus, $f()$ extrahiert die im Korpus nachfolgenden Einheiten. Die zu verkettenden Kandidaten sind durch eine rote Linie verbunden und die potenzielle Konkatenationsstelle ist durch eine vertikale, gestrichelte Linie gekennzeichnet. Die Einheiten einer Kandidatensequenz sind mit durchgehenden Rechtecken umgeben, während ihre Kontexteinheiten im Korpus eine gestrichelte Umrandung besitzen. Zusätzlich sind die im Korpus aufeinanderfolgenden Einheiten von einer Ellipse umschlossen.

Die Auswahl der optimalen Folge von Kandidaten anhand der Einheiten- und Transitionskosten erfolgt durch Minimierung folgender allgemeiner Formel (Black und Campbell, 1995; Campbell und Black, 1996), die auch heute wohl noch die Grundlage der meisten Unit-Selection-Systeme bildet, wenn sich auch verwendete Einheitentypen, Vorauswahl und Kostenfunktionen teilweise deutlich unterscheiden:

$$\sum_{i=1}^{n}(Dc(u_i, u_i - 1) * WC + Du(u_i, t_i) * WU) \tag{2.2.1}$$

Die Variablen u und t sind hier Merkmalsvektoren der in Abbildung 2.2.1 gezeigten Kandidaten- und Zieleinheiten. n steht für die Anzahl der Einheiten in der Zieläußerung, Dc und Du sind Funktionen zur Berechnung der Transitions- resp. Einheitenkosten und WC und WU stellen Vektoren mit Merkmalsgewichten für diese beiden Kostenarten dar. Die Minimierung der Gleichung, also die Ermittlung des optimalen Pfades durch die Kandidatenfolge, wird durch den Viterbi-Algorithmus (Viterbi, 1967) durchgeführt.

Expliziter, indem sie die Aufaddierung der einzelnen Kostenterme (*subcosts*) darstellt, ist die spätere Formalisierung von Hunt und Black (1996) für die Kalkulation der optimalen Bausteinfolge $\bar{u}_1^n$:

$$\bar{u}_1^n = min_{u_1,\ldots,u_n} \sum_{i=1}^{n}\sum_{j=1}^{p} w_j^t C_j^t(t_i, u_i) + \sum_{i=2}^{n}\sum_{j=1}^{q} w_j^c C_j^c(u_{i-1}, u_i) + C^c(S, u_1) + C^c(u_n, S) \tag{2.2.2}$$

Hier sind w^t und C^t die Gewichts- und Kostenvektoren für die Einheitenkosten und w^c und C^c jene für die Transitionskosten. In Erweiterung von Formel 2.2.1 werden auch die Übergangskosten an Satzbeginn $C_j^c(u_{i-1}, u_i)$ und -ende $C^c(S, u_1)+C^c(u_n, S)$ hinzugerechnet.

Auch wenn in den frühen Stadien von CHATR eine übermäßige Gewichtung der prosodischen Eignung beizeiten sogar die Lautidentität zwischen Zieläußerung und ausgewählten Bausteinen gefährdete (Möbius, 2000), markierten diese Erweiterungen und die Demonstration der multilingualen Anwendbarkeit wohl den Anfang des Paradigmenwechsels hin zur Unit-Selection-Synthese. Zwar hatte bereits Jahre zuvor, kurz nach der Publikation von Sagisaka (1988), an den NTT Human Interfaces Laboratories eine parallele Entwicklung der Unit-Selection begonnen, die ebenfalls prosodische Kriterien wie durchschnittlicher F0-Wert, Grundfrequenzkontur, Dauer und Amplitude bei der Auswahl von (uniformen) Phon-Einheiten berücksichtigte (Hirokawa, 1989; Hirokawa und Hakoda, 1990). Offenbar wurde jedoch nie versucht, dieses Verfahren auf andere Sprachen als das Japanische anzuwenden. Auch standen zum Zeitpunkt der Veröffentlichung im Mittel noch deutlich weniger Rechenleistung und Speicherplatz zur Verfügung als bei der Vorstellung des CHATR-Ansatzes. Vermutlich können diese Umstände für die späte Durchsetzung des Unit-Selection-Ansatzes verantwortlich gemacht werden.

Das CHATR-Syntheseverfahren war auch eine der Inspirationsquellen für die Syntheseaktivitäten am Institut für Kommunikationsforschung (IKP)[3] ab Ende der neunziger Jahre. Ermutigt durch erste Versuche zur Einheitenauswahl vom Korpus durch Portele (1998) wurde die Synthese des Verbmobil-Projekts zur *speech-to-speech translation* durch ein Unit-Selection-Verfahren ersetzt. Nachfolgend wird das resultierende System beschrieben. Es stellt die Grundlage für das BOSS-System dar, dessen Entwicklung im Zentrum dieser Arbeit steht (s. Kapitel 4 und 5).

2.3. Die Verbmobil-Synthese

Verbmobil war ein vom deutschen Bundesministerium für Bildung und Forschung gefördertes Projekt zur Entwicklung eines Systems, das natürliche gesprochene Sprache in eine synthetische sprachliche Äußerung in einer anderen Sprache übersetzen kann (Wahlster, 2000), um verschiedensprachigen Sprechern die Absprache von Terminen zu ermöglichen. Während der achtjährigen Förderungsdauer von 1993 bis zum Jahr 2000 waren insgesamt 31 Partnerorganisationen aus Wissenschaft und Industrie beteiligt (Karger und Wahlster, 2000). Angestrebte Quell- und Zielsprachen für die Übersetzung waren Deutsch, Englisch und Japanisch. Die hier geschilderte Entwicklung der deutschen Unit-Selection-Synthese stellt dabei den für die vorliegende Arbeit relevanten Beitrag des IKP zum finalen Verbmobil-System dar.

Anfänglich basierte die Synthese im Verbmobil-Projekt auf dem traditionellen konkatenativen Ansatz für jeden Einheitentypen nur eine — prosodisch möglichst ,,neutrale" — Variante vorzuhalten und diese dann durch intensive Signalmanipulation an die Zielspezifikation anzupassen. Die verwendeten Einheiten waren Halbsilben, Diphone, Präfixe und Suffixe nach der MIX-Inventardefinition von Portele (1996). Wagner et al. (1999) berichten, dass dieses Verfahren gegen Ende des Projektes in die Kritik geriet, ,,hauptsächlich aufgrund des Mangels an Natürlichkeit". Diese auf Basis des Diphonansatzes weiter steigern zu können schien jedoch aussichtslos. Schließlich ist die Motivation jeglicher datenbasierter, konkatenativer Verfahren, fehlendes Wissen über die Sprachproduktion durch die Nutzung der inhärenten Eigenschaften von Sprachbausteinen zu ersetzen, so dass eine explizite, akustisch detaillierte Modellierung entfallen kann (vgl. auch die Diskussion in Taylor und Black, 1999). Die traditionelle Diphonsynthese deckt dabei aufgrund ihrer 1:1-Relation zwischen Einheiten für einen bestimmten Laut, ein Diphon, etc. und dessen Instanzen bestenfalls aber nur die spektrale Variabilität ab, die aufgrund einer abstrakten Verallgemeinerung des Lautkontextes auftritt. Jegliche prosodische, sprecher-

[3]— jetzt Abteilung *Sprache und Kommunikation* am *Institut für Kommunikationswissenschaften* (IfK) —

und äußerungskontextabhängige Variabilität muss modelliert und das Sprachsignal gemäß den Vorhersagen des Modells modifiziert werden. Da aber damals wie heute weder die Modelle noch die Manipulationsmethoden annähernd in der Lage waren und sind, natürliche Ausgaben zu erzeugen, blieb zur Verbesserung der Verbmobil-Synthese nur die Möglichkeit, auf längere und variablere natürlichsprachliche Einheiten zurückzugreifen.

Ausgangspunkt für die Entwicklung des neuen Ansatzes war nach Stöber (2002) die Beobachtung, dass sogenannte reproduktive Synthesen (Hess, 2002) in der Bewertung der Natürlichkeit gut abschneiden. Diese Systeme arbeiten nach dem Prinzip des *phrase slot filling*, bei dem in einem oder mehreren immer wiederkehrenden aufgezeichneten Sätzen und Phrasen nur einzelne Wörter ausgetauscht werden. In den besseren Ausführungen sind diese Wörter prosodisch auf den Zielkontext optimiert, indem sie einer äquivalenten Phrasenposition enstammen oder diese durch ihre Realisierung simulieren. Beispiele für solche Systeme geben Ansagen an Bahnhöfen, in denen nur eine größere feststehende Menge von Zeiten, Zielen und Zugnamen in einer kleinen Auswahl verschiedener Satzkontexte eingesetzt wird. Auch in Verbmobil war das überwiegend genutzte Vokabular auf eine Menge von ca. 17.000 Wörtern begrenzt. Zudem war durch die Festlegung auf die Terminplanungsaufgabe auch eine Vielzahl wiederkehrender Phrasen in den Dialogen zu beobachten. Außer letzteren mussten aber auch unbekannte Eingabesätze synthetisiert werden können, daher schied eine reine reproduktive Synthese als Methode aus. Stöber et al. (1999) entwickelten daher einen hybriden Ansatz, der eine Auswahl und Konkatenation ganzer Wörter nach den grundlegenden Prinzipien der CHATR-Synthese verwendete. Unbekannte Wörter, wie die häufig vorkommenden Eigennamen, wurden zunächst noch durch die alte Diphonsynthese realisiert. Um die daraus resultierenden großen Unterschiede in Qualität und Klangcharakteristik einzudämmen und bestärkt durch die guten Evaluationsergebnisse der Wortsynthese wurde das neue Syntheseverfahren später um eine optionale Phonebene erweitert. Im Zusammenspiel mit eigens aufgenommenen Korpora wurde die Verbmobil-Synthese damit zu einem Multilevel-Unit-Selection-System (Stöber et al., 2000), dessen Einheitenauswahlverfahren in den wichtigen Aspekten auch dem des später entwickelten BOSS-Systems entspricht.

2.3.1. Das Einheitenauswahlverfahren der Verbmobil-Synthese

Der eigentlichen Einheitenauswahl in Verbmobil ist eine sogenannte Pre-Selection oder Vorauswahl vorgeschaltet. Ihre Funktion ist einerseits, die Menge der Kandidaten für eine Einheit zu begrenzen, indem sie nur solche Units aus der Datenbank für die Auswahl zulässt, deren symbolisch annotierte Eigenschaften mit denen des Ziels übereinstimmen. Andererseits entscheidet sie auch, welcher Einheitentyp für

einen bestimmten Äußerungsabschnitt zum Einsatz kommen soll. Dabei wird versucht, immer den größtmöglichen Einheitentypen zu wählen. Begonnen wird mit dem Wort und einem bestimmten Satz von Kriterien. Genügt keiner der Kandidaten in der Datenbank diesen Anforderungen, kann eine weitere Suche mit weniger strengen Auflagen gestartet werden. Dies setzt sich solange fort, bis entweder eine Ergebnismenge aus der Datenbank zurückgeliefert wird oder aber alle vordefinierten Suchanfragen für die Ebene erfolglos beendet sind. In letzterem Falle wird versucht, den Äußerungsabschnitt aus Einheiten der nächstkleineren Ebene zusammenzusetzen. Der Vorgang wird solange weitergeführt, bis auf einer Ebene Einheiten gefunden werden, die den Anforderungen entsprechen. Diese werden an die Einheitenauswahl übergeben.

Obwohl offenbar konzeptuell weitere Auswahlebenen angedacht waren (Stöber et al., 2000), werden in der Verbmobil-Synthese nur Worteinheiten mit Fallback auf die Phonebene verwendet (Stöber, 2002). Als Kriterien für den Rückgriff auf die unterste Ebene geben die Autoren die Nichtübereinstimmung der orthographischen Repräsentation eines Wortes sowie des phonetischen Kontextes mit der Zielvorgabe an. Ob die Auswahl innerhalb der Suche auf der Wortebene dabei gestuft ist oder bei Nichterfüllung aller Kriterien direkt auf Phoneinheiten zurückgegriffen wird, bleibt zunächst offen. Angesichts des unten geschilderten Koartikulationskriteriums für die Kostenbewertung ergibt allerdings nur die erste Variante einen Sinn, so dass wohl auch Worteinheiten ohne adäquaten Zielkontext gewählt werden. Auch zu den Auswahlkriterien der Phonebene werden keine Angaben gemacht. Hier könnte z. B. analog dem später in BOSS II verwendeten Verfahren (s. Abschnitt 4.2.4) zunächst nach Einheiten im exakten phonetischen Kontext gesucht werden. Bei Rückgabe einer leeren Treffermenge würden dann alle Instanzen des Zielphons als Kandidaten für die Unit Selection ausgewählt. Im Gegensatz zur Vorauswahlstufe bei CHATR, dem Pruning auf 20 bis 50 Kandidaten nach Höhe der Einheitenkosten, werden bei BOSS und Verbmobil nur kategoriale Merkmalsunterschiede zwischen Kandidaten und Zieläußerung berücksichtigt, so dass sich die Vorauswahl nicht für akustische Maße und andere kontinuierliche Werte eignet (s. auch Abschnitt 6.3.3).

Die in der Pre-Selection ausgewählten Kandidaten für jeden zu synthetisierenden Abschnitt eines Zielsatzes können in einem Graphen dargestellt werden. Abbildung 2.3.1 zeigt einen solchen Graphen für die Äußerung ,,Neues Beispiel". Die Spalten repräsentieren die einzelnen Äußerungsteile und die für ihre Synthese zugrundegelegten Einheiten. Die Zeilen listen je Spalte einen aus der Vorauswahl hervorgegangenen Kandidaten in Form eines Knotens des Graphen. Alle Kandidaten für einen bestimmten Äußerungsabschnitt müssen vom selben Einheitentyp sein, um die Anzahl der Spalten konstant zu halten. Die Möglichkeit, überhaupt verschiedene Einheiten in einem Graphen zu verbinden, ist dabei eine Eigenschaft des aus Verbmobil her-

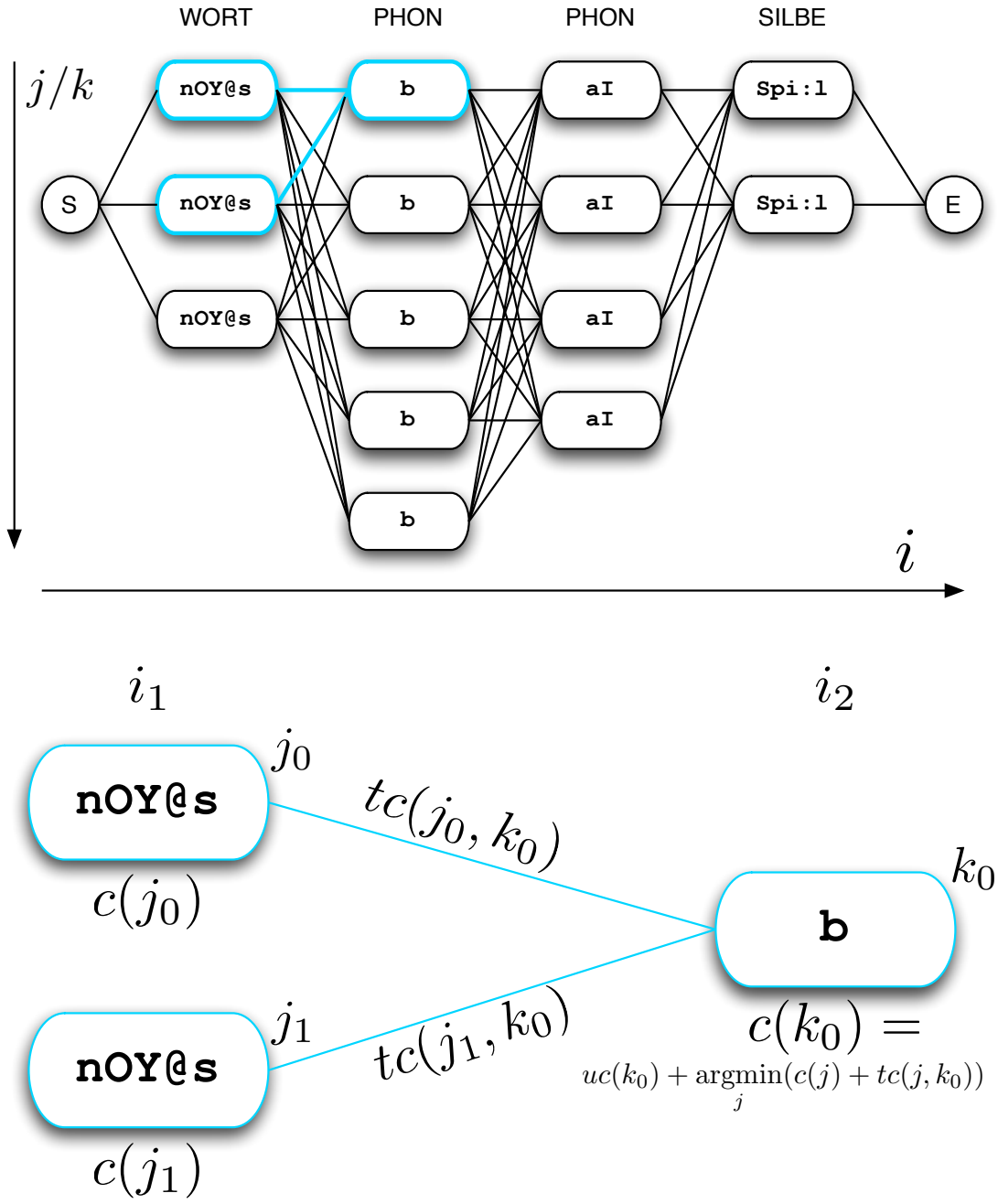

Abbildung 2.3.1.: Graphendarstellung der Einheitenkandidaten (oben) und rekursive Berechnung der optimalen Einheitenfolge (unten) in BOSS/Verbmobil.

vorgegangenen BOSS-Systems, die so in der Verbmobil-Synthese nicht vorhanden ist. Dort werden fehlende Wortkandidaten durch virtuelle Wortknoten ersetzt und die Phonketten zur Synthese dieser Wörter in separaten Graphen dargestellt und berechnet. Silben finden in der Verbmobil-Synthese keine Verwendung. Die zusam-

menfassende Darstellung konnte hier gewählt werden, weil die Berechnung in beiden Systemen ansonsten identisch ist.

Für jede Spalte findet, entsprechend der oben angesprochenen Vorauswahl bei CHATR, ein Pruning der Kandidatenmenge auf eine konstante Höchstanzahl statt, so dass nur die Knoten mit den niedrigsten Einheitenkosten verbleiben. Jeder Kandidatenknoten einer Spalte ist mit allen Kandidaten der Vorgänger- und Nachfolgerspalten durch Kanten verbunden, und jede solche Kante steht für eine mögliche Konkatenationsstelle und die damit verbundenen Transitionskosten. Letztere entsprechen, wie auch die Einheitenkosten, konzeptuell und in ihrer Verwendung in der Auswahl den in CHATR verwendeten Kostentypen, wenn auch die einzelnen Kostenterme oder *subcosts* zu einem großen Teil differieren.

Um nun die beste Folge von Kandidaten für die Zieläußerung zu finden, werden für jeden Kandidaten k einer Spalte i auf folgende rekursive Weise die Gesamtkosten c für den bis dorthin günstigsten Pfad durch den Graphen berechnet und im Kandidatenknoten gespeichert:

$$c(k_0) = uc(k_0) + \underset{j}{\operatorname{argmin}}(c(j) + tc(j, k_0)) \tag{2.3.1}$$

Zunächst wird also der Knoten j der Vorgängerspalte $i - 1$ gesucht, für den die Summe des gespeicherten Kostenwerts und der Transitionskosten tc zum Knoten k_0 minimal ist. Die Summe wird zu den Einheitenkosten uc von k_0 addiert und in dessen Knoten gespeichert. Der gefundene Kandidat der Spalte $i-1$ wird als bester Vorgänger für k_0 markiert. Da der Kostenwert für diesen Vorgängerknoten ebenfalls nach Formel 2.3.1 berechnet wurde[4], steht damit bereits der beste Pfad bis Knoten k_0 fest. Dieser Vorgang setzt sich für alle Kandidaten der Spalte i fort und wird dann für jede folgende Spalte wiederholt. Mit Erreichen des Endknotens steht der kostengünstigste Pfad durch den Graphen fest. Der untere Teil von Abbildung 2.3.1 illustriert die Berechnung der Kosten eines Kandidaten k_0.

Die Berechnung der Einheiten- und Transitionskosten in der Verbmobil-Synthese erfolgt nach den Gleichungen 2.3.2 und 2.3.3.

$$uc(k) = \frac{1}{n} \sum_{s=1}^{n} w_s^u c_s^u(k) \tag{2.3.2}$$

$$tc(j, k) = \frac{1}{m} \sum_{s=1}^{m} w_s^t c_s^t(j, k) \tag{2.3.3}$$

[4]Die einzigen Ausnahmen stellen die Start- und Endknoten des Graphen dar, deren Kostenwert immer 0 beträgt.

Die einzelnen *subcost*-Terme sind durch den Vektor c^u repräsentiert; w^u sind die dazugehörigen Gewichte. Analog sind c^u und w^u für die Transitionskosten definiert. Damit entspricht die Berechnung der Kosten bis auf die Division durch die Gesamtzahl der Unterkostenfunktionen einer Kostenart jener in CHATR (s. Abschnitt 2.2). Die in Verbmobil genutzten Unterkostenterme für die Transitions- und Einheitenkostenkalkulation werden im Folgenden aufgelistet.

Fünf Einheitenkosten- und zwei Transitionskostenterme werden in Verbmobil für die Auswahl von Wörtern verwendet. Auf der Phonebene ist das Verhältnis umgekehrt. Für Wortknoten gehen die Abweichung der Eigenschaften Wortposition in der Phrase, Phrasentyp, Wortklasse und Dauer zwischen Ziel und Kandidat sowie die Reduziertheit des Wortes in die einheitenbezogenen Kosten ein. Mit Reduktion wird dabei jegliche wahrnehmbare Abweichung von der kanonischen Einzelwortaussprache bezeichnet. Unabhängig vom Grad der Abweichung werden reduzierte Wörter mit einem festen Kostenwert belegt. Verkettungskosten werden auf Basis der vorhandenen oder nicht vorhandenen Konsekutivität zweier Einheiten im Korpus vergeben. Zusätzlich wird geprüft, ob der letzte Laut des Kandidaten j mit dem Laut übereinstimmt, der dem folgenden Kandidaten k in seinem Korpuskontext vorangeht. Umgekehrt wird auch der erste Laut von k mit dem rechten Korpuskontext des Wortkandidaten j verglichen. Ergeben beide Tests identische Umgebungen, werden keine Kosten zugewiesen. Stimmen beide Umgebungen nicht überein, sind die Kosten maximal. Ist nur ein Kontext unterschiedlich, hängt der Wert davon ab, welche Seite betroffen ist. Dabei werden der persistenten Koartikulation gravierendere Auswirkungen zugeschrieben als der antizipatorischen (vgl. O'Shaughnessy, 1987). Dementsprechend wird ein negatives Ergebnis beim Vergleich zwischen j und dem linken Korpuskontext von k mit höheren Kosten belegt. Dieser Vorgang ist nicht mit den in Abschnitt 2.2 beschriebenen Beurteilungen des Korpuskontextes im Rahmen der Transitionskostenberechnung von CHATR zu verwechseln, die auch einheitenbezogene akustisch-prosodische Abweichungen misst. In Verbmobil werden ausschließlich die segmentalen Lautbeschreibungen auf Identität oder Ungleichheit geprüft.

Die Einheitenkosten für die Lautauswahl werden über die Bestimmung der Unterschiede zwischen den Zielvorgaben für die Dauer und die Position des Lautes im Wort und den entsprechenden Eigenschaften der Kandidateneinheiten ermittelt. Transitionseigenschaften der Phone werden durch die akustischen Maße F0- und Energie-Unterschied sowie die bereits für die Wortauswahl beschriebenen Kriterien Konsekutivität und Koartikulation bewertet. Zusätzlich wird für nicht konsekutive Laute untersucht, ob die Anregungsart an der potenziellen Konkatenationsstelle von stimmhaft zu stimmlos oder umgekehrt wechselt. Solche Übergänge verursachen geringere spektrale Störungen und werden daher in der Auswahl bevorzugt.

Durch die unterspezifizierte Intonation, die sich auf die Auswahl von Kandidaten in einer passenden Phrasenposition beschränkt, beruht die Verbmobil-Unit-Selection zumindest in Teilen auf denselben Prämissen wie das *phonological structure matching*, PSM (Taylor und Black, 1999; Taylor, 2000), auch wenn dieser Umstand möglicherweise dem engen Zeitrahmen des Projektes geschuldet ist. Bei diesem Auswahlansatz wird auf die Berechnung expliziter akustischer Vorgaben für die prosodischen Parameter zunächst verzichtet, da diese häufig fehlerbehaftet sind. Stattdessen wird versucht, möglichst lange passende Äußerungsteile anhand der Phrasierung und phonologischen Struktur des Ziels auszuwählen und deren implizite prosodische Eigenschaften zu nutzen. Anders als bei Verbmobil wird diese Struktur nach den Konventionen der metrischen Phonologie dargestellt (s. Abbildung 2.5.2), und es können Folgen bzw. nicht-uniforme Einheiten vom Phon bis hin zu ganzen Phrasen oder Sätzen ausgewählt werden. Für alle Fälle, die nicht vom Korpus abgebildet werden, existiert beim PSM ein Fallback auf die Methoden der klassischen konkatenativen Synthese, also auf Prädiktion akustischer Parameter und anschließende prosodische Manipulation. Auch hier unterscheidet sich die Verbmobil-Synthese, indem sie zwar von Anfang an aufgrund vorhergesagter Dauerwerte auswählt, jedoch keine Manipulation des Sprachsignals vornimmt. Beide Ansätze eint, dass sich die Unterspezifikation auch auf die segmentale Beschreibung erstreckt, ein Umstand, der in Kapitel 3 diskutiert wird.

Möbius (2000) kritisiert die Auswahl auf Wortebene, weil sie auf der falschen Annahme beruhe, dass die Koartikulation über Wortgrenzen hinweg schwächer sei. Die Motivation in Verbmobil lag aber wohl eher in dem Versuch, möglichst lange Einheiten auszuwählen und dabei die Auswahlkomplexität aus Laufzeitgründen zu verringern. Die Auswahl von Wörtern zur Erreichung dieses Ziels gründet sich dabei auf die Genese der Verbmobil-Synthese aus dem *phrase-slot-filling*. Dabei wird aber, wie auf der Phonebene auch, über die Einheitengrenze hinaus versucht, durch Einsatz der Konsekutivitäts- und Koartikulationskriterien möglichst lange zusammenhängende Ketten zu finden oder zumindest einen möglichst ähnlichen Kontext zu wählen.

Die Ansätze des Verbmobil-Systems wurden später in der BOSS-Synthese weiterentwickelt, deren grundlegende Architektur in Kapitel 4 erläutert wird. Die eigenen Arbeiten zur Erweiterung des BOSS-Systems werden im darauf folgenden Kapitel 5 vorgestellt.

2.4. Weitere Systeme und Methoden

Auf die Konzepte der multilingualen CHATR-Architektur bauend, wurde am Centre for Speech Technology Research, CSTR, in Edinburgh das Synthesesystem Festival entwickelt (Black und Taylor, 1997b; Taylor et al., 1998), das einerseits zur reinen Diphon-Synthese genutzt werden konnte, andererseits aber auch Unit Selection mit einer Abwandlung des CHATR-Verfahrens erlaubte (Black und Taylor, 1997a). Festival wurde damit schnell zur Grundlage vieler Syntheseaktivitäten weltweit. Bei AT&T schuf man aus der Festival-Architektur, den symbolverarbeitenden Komponenten der Bell-Labs-Synthese Flextalk und der ursprünglichen Unit-Selection von CHATR das AT&T Next-Gen TTS System (Beutnagel et al., 1999), das als kommerzielles System unter dem Marketingnamen NaturalVoices (AT&T Intellectual Property, 2007) nach der Veröffentlichung im Jahr 2002 lange Zeit als eines der besten TTS-Systeme gehandelt wurde. Andere erfolgreiche Unit-Selection-Synthesen auf Festival-Basis waren die kommerziellen Produkte von Rhetorical Systems (rVoice) und Cepstral. Auch in der universitären Forschung wurde und wird Festival häufig eingesetzt und an verschiedene Sprachen angepasst (z. B. Möhler et al., 2007). Am CSTR setzten Clark et al. (2007) die Entwicklung von Festival fort und bauten die Unit-Selection-Komponente weiter aus.

Parallel zu den Entwicklungen, die auf den ν-Talk- und CHATR-Konzepten fußten, forschten die Sprachtechnologie-Labore von IBM (Donovan und Eide, 1998; Donovan und Woodland, 1999) und Microsoft (Huang et al., 1996) an Möglichkeiten, mit der Anwendung von Hidden-Markov-Modellen, HMM (Rabiner, 1989), statistische Methoden aus der Spracherkennung auf die Synthese zu übertragen. Da auch bei diesen Verfahren aus mehreren Einheiten, repräsentiert durch Zustände (*states*) der HMM, nach deren Eigenschaften gewählt werden kann, sind auch sie dem Unit-Selection-Paradigma zuzuordnen. In jüngerer Zeit erfreut sich die HMM-basierte Synthese wachsender Aufmerksamkeit, gefördert durch die Entwicklung des HTS-Systems am Nagoya Institute of Technology (Tokuda et al., 2004; Zen et al., 2007). Die Besonderheit von HTS liegt darin, dass nicht nur die Auswahl durch HMM-Techniken realisiert wird, sondern auch die Sprachgenerierung auf der Basis von Modellzuständen erfolgt, wodurch eine sehr kompakte Repräsentation von Sprachkorpora realisiert werden kann. Eine Anpassung an das Deutsche auf Basis des Verbmobil-Korpus stellen Weiss et al. (2005) vor. Qualitativ reicht das HTS-Verfahren jedoch nicht an die kostenbasierten Unit-Selection-Methoden heran. Da die in der vorliegenden Arbeit präsentierten Ergebnisse in der Tradition der Einheitenauswahl mit linguistisch-phonetisch motivierten Einheiten stehen, soll diese Thematik hier nicht weiter vertieft werden. Die nachfolgenden Abschnitte in diesem Kapitel werden sich darauf konzentrieren, einen Überblick über die historisch und aktuell wichtigsten Fragestel-

lungen des hier verfolgten Unit-Selection-Ansatzes zu geben.

2.5. Forschungsschwerpunkte der Unit-Selection-Synthese

Bereits in Abschnitt 2.1 wurde deutlich, dass eine schier unüberschaubare Zahl von Möglichkeiten existiert, die Auswahl aus multiplen Instanzen einer Syntheseeinheit zu gestalten. In Kombination mit den Optionen zur Definition von Korpora und zur Wahl der Basis-Einheiten von subsegmentalen Units bis hin zu ganzen Phrasen, den Einflüssen verschiedener Messverfahren und Gewichtungen von Kosten oder auch des nicht zu unterschätzenden Eignungsgrades des Sprechers steigt die Komplexität derart an, dass der Beitrag einzelner Design-Entscheidungen oft nur mit aufwändigen Verfahren zu messen ist. Systemübergreifende Vergleiche sind selten und betreffen meist nur die Gesamtsysteme (z. B. Fraser und King, 2007). Noch schwieriger ist es, das Zusammenspiel verschiedener Methoden in unterschiedlichen Systemen zu messen (s. auch die Diskussion in Abschnitt 4.2.1). Im Folgenden werden die verschiedenen Ansätze diskutiert, einige der genannten Teilprobleme der Unit Selection zu lösen. Aufgrund der geschilderten Schwierigkeit der Evaluation darf jedoch keine konklusive Antwort auf die Frage erwartet werden, welche Methode nun die jeweils beste darstellt.

2.5.1. Auswahlalgorithmen und prosodische Manipulation

Allen Verfahren der ATR-Labs bis hin zur ersten Version von CHATR war gemein, dass sie wie die klassische konkatenative Synthese eine prosodische Manipulation des Sprachsignals entsprechend den Zielvorgaben für Dauer und Grundfrequenz vornahmen, obwohl bei CHATR durch Berücksichtigung dieser Parameter in der Einheitenauswahl die Unterschiede zwischen Wunsch- und Syntheseprosodie reduziert werden konnten. Campbell (1996) geht daher später einen anderen Weg in der Weiterentwicklung von CHATR und plädiert für die Nutzung größerer, variantenreicherer Korpora bei gleichzeitigem Verzicht auf jegliche Signalmanipulation. Auch in der in Festival genutzten Abwandlung der CHATR-Einheitenauswahl von Black und Taylor (1997a) wird zunächst keine prosodische Manipulation vorgenommen. Die Autoren argumentieren allerdings, dass eine minimale Signalmanipulation sinnvoll sei, wenn adäquate Einheiten in der Datenbank fehlen. In dem *clunits* genannten Verfahren ist die Vorauswahl stark erweitert und bezieht außer der Lautidentität und dem segmentalen Kontext auch Einheitencharakteristika wie Dauer und Grundfrequenz mit ein. Kandidateneinheiten, im Unterschied zu CHATR uniforme Phone, werden dabei

in Klassen (*cluster*) eingeteilt, die als Blätter in einem CART-Entscheidungsbaum (*classification and regression trees*, Breiman et al., 1984) angeordnet sind. Bei der automatischen Erstellung dieser Bäume werden Fragen zu den Eigenschaften der Einheiten verwendet, um Klassen zu bilden und aufzuteilen. Eine Aufteilung erfolgt, wenn eine Frage die mittlere akustische Distanz zwischen allen Einheiten der daraus entstehenden Unterklassen gegenüber der zusammengefassten Klasse reduzieren (s. Abbildung 2.5.1) und jede neue Klasse mindestens 10 bis 20 Kandidaten enthalten würde.

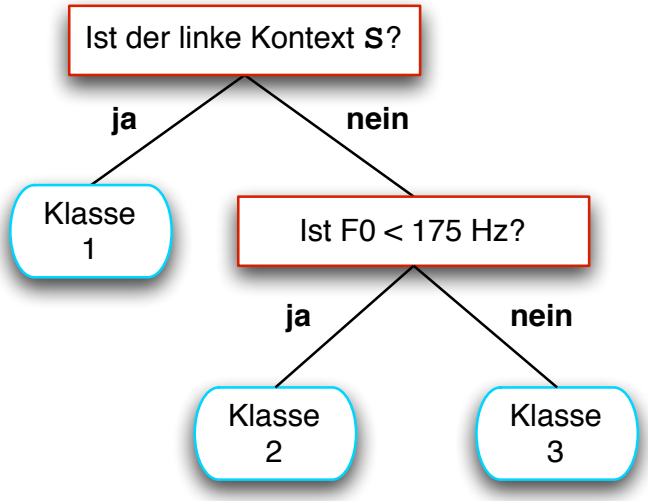

Abbildung 2.5.1.: Hypothetischer Ausschnitt eines Entscheidungsbaumes zum Clustering von Syntheseeinheiten.

Die akustische Distanz *Adist* (Gleichung 2.5.1) ist dabei definiert als mittlere Summe der gewichteten Differenzen der Parameter MFCC, Delta-Cepstrum, F0 und Intensität zwischen jedem Fenster j zweier Einheiten U und V. Längenunterschiede der Einheiten werden durch Interpolation von Fenstern in der kürzeren Einheit angeglichen. WD ist hier ein Strafgewicht für den Längenunterschied und SD bezeichnet die Standardabweichung von j.

$$\text{wenn } |V| > |U|, Adist = \frac{WD|U|}{|V} \sum_{i=1}^{|U|} \sum_{j=1}^{n} \frac{W_j |F_{ij}(U) - F_{(i\frac{|V|}{|U|})j}(V)|}{SD_j n|U|} \tag{2.5.1}$$

Das Clustering in einem Entscheidungsbaum erlaubt eine effizientere Pre-Selection gegenüber dem CHATR-Verfahren, weil die Bewertung der Einheiteneignung größtenteils nicht zur Laufzeit stattfinden muss, das Pruning entfällt und die Unit Selection damit auf die Minimierung des folgenden Ausdrucks reduziert werden kann:

$$\sum_{i=1}^{N} Tdist(U_i) + W \times Jdist(U_i, U_i - 1) \tag{2.5.2}$$

Jdist ermittelt hier die Transitionskosten nach dem CHATR-Verfahren. Etwas problematisch erscheint, dass trotz Pre-Selection eine weitere Eignungsbewertung der Einheitenkosten durch die Funktion *Tdist* vorgenommen wird, die den Abstand eines Kandidaten vom Klassenzentrum (nach *Adist*) bewertet. Dies soll möglicherweise verhindern, dass schlecht segmentierte oder uncharakteristische Einheiten ausgewählt werden. Bei einer kleinen Menge von 10–20 Kandidaten pro Klasse ist allerdings fraglich, ob solche Einheiten durch größeren Abstand zum Mittelwert gekennzeichnet sind. Allerdings werden die Transitionskosten in der Auswahl stärker gewichtet als die Einheitenkosten, so dass der Einfluss von *Tdist* begrenzt ist.

Auch Donovan und Eide (1998) verwenden in der IBM-Synthese Clustering zur Vorauswahl. Hier sind es allerdings einzelne HMM-Zustände, die die Blätter der Entscheidungsbäume bevölkern, und die Aufteilung verwendet ausschließlich Fragen zum phonetischen Kontext. F0 und Dauer werden separat prädiziert und fließen in eine Kostenfunktion nach dem CHATR-Muster ein, die alle Signalabschnitte untersucht, die einem Blatt zugeordnet sind. Dadurch ergibt sich eine hohe Zahl von Kandidaten für eine Klasse. Später führt Donovan (2000) daher ein Offline-Pruning der Bäume ein, das er Pre-Selection nennt[5]. Die IBM-Synthese auf dem in Donovan (2000) geschilderten Stand verwendet prosodische Manipulation durch *pitch synchronous overlap add*, PSOLA (Moulines und Charpentier, 1990), um die ausgewählten Segmente an die Zielprosodie anzupassen, versucht aber die Zielvorgaben an die ausgewählten Segmente so anzupassen, dass die Manipulation in einem perzeptiv tolerierbaren Rahmen bleibt.

Während die IBM- und Festival-Ansätze Entscheidungsbäume zur Auswahl uniformer Einheiten (sog. *fenemes* bzw. Phone) verwenden, benutzt das in 2.3.1 geschilderte PSM solche Bäume zur Darstellung metrischer Strukturen, aus denen nicht-uniforme Einheiten bis hin zum Satz ausgewählt werden können. Da es keine akustischen Zielvorgaben gibt, ist eine Manipulation von Dauer und Intonation nicht möglich. Die Argumentation von Taylor und Black (1999) ist dabei, vergleichbar der von Campbell (1996), dass das grundlegende Konzept der Unit Selection die Nutzung inhärenter Signaleigenschaften anstatt modellierter Parameter sei und lange

[5]Da beide Begriffe unterschiedliche Methoden zum gemeinsamen Ziel der Reduktion einer Kandidatenmenge für die Unit Selection subsumieren, sind sie teilweise austauschbar und werden auch so verwendet. In Anlehnung an die Verbmobil- und BOSS-Terminologie soll Pre-Selection oder Vorauswahl im Folgenden als zur Laufzeit oder offline stattfindende Methode der Reduktion nach einheiten- und kontextbedingten Kriterien verstanden werden und Pruning ein Offline-Verfahren bezeichnen, das Einheiten komplett aus der Datenbank entfernt.

zusammenhängende Einheiten daher besser seien für die Qualität. Auch wenn diese Meinung vom Autor der vorliegenden Arbeit geteilt wird (s. auch Kapitel 3) und bei Auswahl von ganzen Phrasenteilen das PSM sicher zu hervorragenden Ergebnissen kommt, ist die Menge an prosodischen Freiheitsgraden, die eine rein metrische Vorgabe auf der Wort- und Subwortebene erlaubt, potenziell problematisch für die Qualität. Die Gefahr besteht, dass ausgewählte Einheiten zwar nicht einer ,,gleichmacherischen" modellierten Prosodie entsprechen, gleichzeitig aber auch mit einer natürlichen Realisierung nichts gemein haben. PSM eignet sich daher möglicherweise besser für die sogenannte geschlossene Domäne, also ein bestimmtes Anwendungsfeld der Synthese mit häufig wiederkehrenden Phrasenstrukturen, wie es das Terminauskunftsszenario von Verbmobil darstellt. Dies sehen auch Schweitzer et al. (2003) so und kombinieren daher in der SmartKom-Synthese auf Festival-Basis das PSM-mit dem Clunits-Verfahren, um in verschiedenen Auskunftsdomänen sowohl für die domänenspezifischen Phrasen als auch für korpusfremde Wörter immer die ideale Methode wählen zu können.

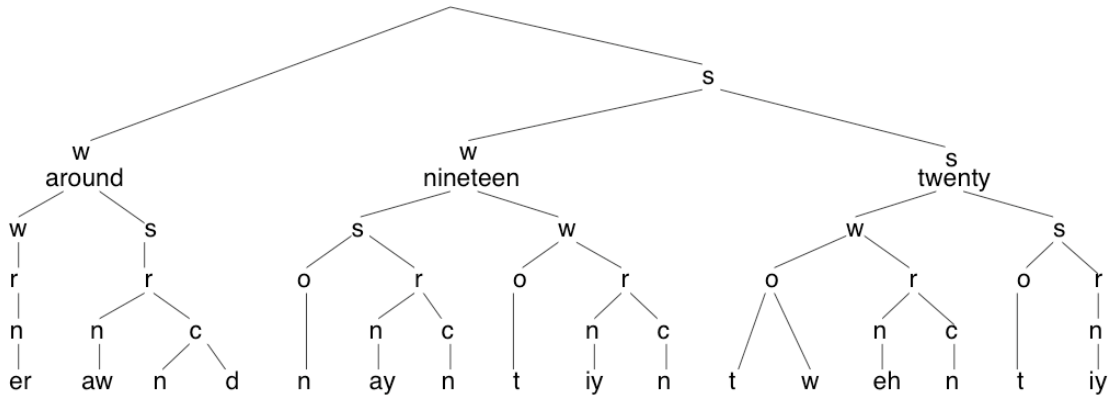

Abbildung 2.5.2.: Metrisch-phonologische Darstellung eines Inventarsatzes als Grundlage für *phonological structure matching* – Taylor und Black (1999).

Das Laureate-System der British Telecom (Breen und Jackson, 1998) verwendet ebenfalls phonologische Kriterien zur Einheitenauswahl, allerdings fast ausschließlich zur Beschreibung der segmentalen Eigenschaften in Form von eigenen distinktiven Merkmalen. Phonologisch und suprasegmental werden nur die Position in der Silbe und der Grad der Akzentuierung annotiert. Zusätzlich wird die Dauer jeder Einheit gespeichert. Einheitenkandidaten werden in der Vorauswahl auf Basis einer Baum-Struktur aller Phone des Korpus zu nicht-uniformen Einheiten einer fixen Maximalgröße zusammengesetzt.

Auch im Verfahren von Carvalho et al. (2003) wird das gesamte Korpus in einem gerichteten Graphen dargestellt, genauer in einem endlichen gewichteten Transduktor oder WFST (*weighted finite state transducer* Salomaa und Soittola, 1978), in

dem alle konsekutiven Einheiten und zusätzlich alle Einheiten mit gleicher segmentaler Beschreibung durch Kanten verbunden sind. Auch die Zieläußerung wird als WFST kodiert und mit dem Korpus-Transduktor komponiert, so dass der resultierende Graph nur noch diejenigen Folgen enthält, aus denen die Äußerung zusammengesetzt werden kann. Für alle Folgen wird ein kleiner Satz von Transitions- und Einheitenkosten berechnet und in den Kantengewichten gespeichert. Auf dieser Basis wird der beste Pfad ausgewählt. Das Hauptproblem des Ansatzes ist nach Carvalho et al., dass die Berechnung aller Folgen zur Laufzeit sehr zeitintensiv ist, gleichzeitig aber die Offline-Berechnung und Einbindung der Konkatenationskosten in den Graphen aus speichertechnischen Gründen unmöglich wäre.

Ein weiteres am IfK entwickeltes Verfahren nutzt statistische Methoden zur Einheitenauswahl (Weiss, 2007; Weiss und Hess, 2006). Die in der audiovisuellen Synthese AVISS implementierte, an Verbmobil angelehnte Multilevel-Unit-Selection mit hierarchisch angeordneten Wort-, Silben-, Diphon- und Phoneinheiten basiert auf bedingten Zufallsfeldern (*conditional random fields*, CRF, Lafferty et al., 2001). Mit diesen kann ermittelt werden, mit welcher Wahrscheinlichkeit eine bestimmte Zieleinheit (die Beobachtung) in Abhängigkeit von den vorher ausgewählten Einheiten von einem Kandidaten erzeugt wird. Maßgeblich sind dafür Merkmalsvektoren der Ziel- und Korpuseinheiten, die denen in der kostenbasierten Einheitenauswahl entsprechen. Perzeptionsexperimente zum Vergleich der statistischen Methode mit einer nicht näher spezifizierten einheiten- und transitionskostenbasierten Auswahl zeigten vielversprechende Ergebnisse.

Wie auch die letzten geschilderten Ansätze verwenden die meisten aktuellen Synthesen nach Literaturlage offenbar keine prosodische Manipulation des Sprachsignals mehr, nachdem auch PSOLA-Alternativen wie *Harmonics-plus-Noise* (Stylianou, 2000) zwar teilweise bessere Ergebnisse liefern, aber immer noch schwer abschätzbare Störungen induzieren. Es fehlt einfach an geeigneten perzeptiven Maßen, um a priori beurteilen zu können, wie sich ein bestimmter Grad der Manipulation auf eine gegebene Einheit oder Konkatenationsstelle auswirkt. Ein ähnliches Problem ergibt sich bei der Einstellung der Gewichte für Einheiten- und Transitionskosten, auf die die meisten der oben geschilderten Verfahren angewiesen sind. Da die Anzahl potenziell sinnvoller Gewichtungsparameter schon bei einer geringen Anzahl von Kostentermen sehr hoch ist, kann auf subjektive Evaluation durch menschliche Hörer nur sehr eingeschränkt zurückgegriffen werden. Eine maschinelle Evaluation der Syntheseausgabe nach perzeptiven Kriterien wäre daher wünschenswert.

2.5.2. Abstandsmaße, Signalrepräsentation und Kostengewichte

Schon Black und Campbell (1995) sind sich bewusst, dass ein objektives Kriterium zur Einstellung von Gewichten benötigt wird. In dem von ihnen verwendeten Verfahren messen sie die euklidischen Abstände der MFCC-Folgen von mit verschiedenen Gewichtseinstellungen resynthetisierten Äußerungen zu denen des vorher aus dem Korpus entnommenen Originalsatzes[6]. Dies wiederholen sie für verschiedene Sätze und wählen die Gewichtseinstellungen, die sich für eine große Anzahl von Sätzen bewährt haben. Obwohl diese Art des Trainings eine Verbesserung der Synthesequalität mit sich bringt, bemerken Black und Campbell, dass die Mel-Cepstrum-Koeffizienten die menschliche Perzeption nur unvollkommen nachbilden, weil impulsartige Störungen herausgemittelt werden, die sich auf die Wahrnehmung verheerend auswirken können.

Dies führt direkt zum bereits angesprochenen Problem der verwendeten Signalrepräsentationen oder -transformationen und der damit assoziierten Abstandsmaße oder Metriken. Während erstere bestimmen, wie ein Sprachsignal im Zeitbereich in eine andere Darstellung überführt wird, geben die Maße an, wie der Unterschied zwischen zwei solchen Repräsentationen berechnet wird. Beide Komponenten sind entscheidend, wenn es darum geht, die menschliche Wahrnehmung möglichst realitätsnah zu simulieren. Oft wird auch die Kombination der Abstandsberechnung und Signalrepräsentation als das Maß bezeichnet. Dieser Konvention soll hier gefolgt werden.

Stöber et al. (2001) verwenden die Vektordarstellung des psychoakustisch motivierten Modells PEMO (Dau et al., 1996) zur instrumentellen Beurteilung von resynthetisierten Äußerungen und erzielen gute Ergebnisse in der Ähnlichkeitsbeurteilung durch menschliche Hörer für die nach dem Modell besten Einheitenfolgen im Vergleich zur Originaläußerung. Ihr Ziel ist jedoch nicht, die Gewichtungen zu optimieren, sondern die Eignung von PEMO für die Transitions- und Einheitenkosten zu testen. Während PEMO bei ersteren die MFCC ersetzen soll, schlägt Stöber (2002) für die Einheitenkosten vor, jedem Lauttypen eine Menge prototypischer PEMO-Koeffizienten zuzuweisen und diese zur Auswahl PEMO-annotierter Korpuseinheiten zu nutzen. Eine vergleichende subjektive Evaluation von Transitionsmaßen, die auf den Signalrepräsentationen PEMO, MFCC, Nulldurchgangs- und Amplitudenhistogramm beruhen, bescheinigte PEMO zwar eine hohe Perzeptionsnähe, ergab jedoch auch eine – wenn auch uneindeutige – Präferenz für die Amplitudenhistogramme (Schröder, 2004). Dieses von Kirstein und Stock (1976) definierte Maß setzt grob quantisierte Samplewerte in Beziehung zu ihrer Auftretenshäufigkeit.

Donovan (2001) testet für den Einsatz im IBM-System verschiedene populäre Ma-

[6]Diese Vorgehensweise ist auch als *leave-one-out*-Methode bekannt.

ße, unter anderem das Kullback-Leibler-Maß (Kullback und Leibler, 1951), das den Unterschied zwischen zwei Leistungsspektren misst und in den Experimenten von Klabbers und Veldhuis (1998) für die Messung von Diskontinuitäten am besten abschneidet. Er vergleicht dieses u. a. mit einem selbst entwickelten Maß, sowie dem euklidischen Abstand von logarithmierten FFT-Vektoren, der Itakuro-Saito-Distanz von Power-Spektren (Itakura und Saito, 1968) und mehreren Kombinationen von euklidischen und Mahalanobis-Distanzen (Mahalanobis, 1936) mit verschiedenen Cepstral-Darstellungen. Sein eigenes Maß ist eine Mahalanobis-Distanz zwischen perzeptiv modifizierten MFCC, in dessen Berechnung die Zentroiden der MFCC-Vektoren für den jeweiligen Lautkontext mit eingehen. Diese Kontextabhängigkeit wird über einen Entscheidungsbaum mit Fragen zur phonetischen Umgebung hergestellt. Das neue Maß geht deutlich als Sieger aus der Evaluation hervor, bei einer allerdings sehr niedrigen mittleren Korrelation der Hörerurteile untereinander von 0.34. Mit einem ähnlichen Konzept, das Kontextfragen und Fragen zu akustischen Parametern für das Training des Entscheidungsbaumes nutzt, warten Syrdal und Conkie (2005) auf.

Auch Vepa et al. (2002) merken an, dass es kein einzelnes Maß gibt, das für alle spektralen Übergänge optimal ist und experimentieren daher erfolgreich mit gewichteten Kombinationen aus den euklidischen und Mahalanobis-Distanzen von MFCC, Linienspektren und *multiple centroid analysis coefficients*, MCA (Crowe und Jack, 1987). Diese Erkenntnis und deren Konsequenzen für die Gestaltung der Transitionskosten scheinen den wichtigsten Fortschritt auf dem Gebiet der Abstandsmaße zu markieren. Von Bedeutung sind diese Maße insbesondere für Systeme wie die von IBM, AT&T und Rhetorical, mit denen die letzten drei geschilderten Versuche durchgeführt wurden, denn diese arbeiten mit relativ kleinen Einheiten, wie Diphonen, Halbphonen oder einzelnen Fenstern und produzieren daher viele Konkatenationsstellen. Eine andere Methode der Störungsreduzierung ist die Minimierung der Konkatenationsstellen durch Verwendung größerer, ggf. nicht-uniformer Einheiten, wie in ν-Talk, CHATR oder Verbmobil. Die Definition solcher Einheiten ist auch verbunden mit einem der wichtigsten Aspekte für die Qualität einer Unit-Selection-Anwendung – der Definition und Aufnahme eines Korpus.

2.5.3. Korpusdefinition, Einheiten und Konkatenation

Die Beschaffenheit des Synthesekorpus stellt die oberste Qualitätsschranke für die daraus entwickelte Syntheseanwendung dar. Dabei ist die maximale Qualität der Unit Selection von vorneherein dadurch begrenzt, dass es aus kombinatorischen Gründen nicht möglich ist, ein Korpus tolerierbarer Größe zu schaffen, das auch nur für die bereits bekannten und in der Synthese vorhersagbaren Einflussparame-

ter auf die Variabilität einer Äußerung (und damit der spektralen Variabilität) alle möglichen Varianten eines Segmentes enthält (vgl. van Santen, 1997 und Möbius, 2000). Nichtsdestotrotz ermöglicht die konkatenative Einheitenauswahl von großen Korpora die derzeit beste Approximation an menschliche Sprache. Um eine Datenbasis zu schaffen, die möglichst viele der nach heutiger Vorstellung wichtigen Varianten abdeckt, gibt es verschiedene Herangehensweisen. Die Basis können entweder große bestehende Korpora, wie z. B. Hörbücher sein (Breuer et al., 2006) oder dedizierte Synthesekorpora, die im Studio aufgrund von Textvorlagen oder auch frei in informellen Sprechsituationen erstellt werden. Letzteren Ansatz verfolgt Campbell (2001, 2003) in der Absicht, die Natürlichkeit durch immer größere, inkrementell wachsende Korpora mit möglichst ungestellten Aufnahmen zu verbessern. Aus Echtzeit- und Speicheranforderungen heraus gehen Entwickler kommerzieller Systeme jedoch meist einen anderen Weg und begrenzen die Datenbankgröße, indem sie versuchen, Textvorlagen für Korpusaufnahmen zu schaffen, die pro Satz eine möglichst große prosodische und segmentale Vielfalt beinhalten. Informationstechnisch formuliert gilt es dabei, ein Mengenüberdeckungsproblem (*set covering problem*, SCP) zu lösen. Weil diese zur Klasse der NP-vollständigen kombinatorischen Probleme gehören, wird bei der Korpusdefiniton darauf verzichtet, eine global optimale Lösung zu finden. Stattdessen werden sogenannte *greedy algorithms* eingesetzt, die i.d.R. nur ein lokales Optimum auf Basis des Ausgangspunktes finden; s. (Francois und Boeffard, 2001; van Santen und Buchsbaum, 1997) für Anwendungsbeispiele. Grundlage für die Berechnung sind oft große Korpora von Zeitungsartikeln, die mit automatischen Methoden transkribiert und prosodisch annotiert werden. Jeder Satz erhält eine Punktzahl entsprechend der Menge an enthaltenen neuen Einheitentypen. Dabei können verschiedene segmentale und prosodische Eigenschaften unterschiedlich bewertet werden, je nach geschätzter Wichtigkeit für das Korpus. In diesem Fall handelt es sich bei dem zu lösenden Problem um ein *weighted* SCP. Der Satz mit der besten Punktzahl wird ,,greedy" ausgewählt und die verbleibenden Sätze erneut bewertet. Dieser Vorgang wird wiederholt bis entweder alle gewünschten Varianten im Korpus enthalten sind oder eine Höchstmenge an Sätzen gewählt wurde. Die so entstandene Textvorlage kann dann von einem Sprecher realisiert und aufgezeichnet werden. Hierbei ist darauf zu achten, dass die Realisierung segmental und prosodisch der automatisch annotierten Textgrundlage entspricht, um die Abdeckung nicht zu gefährden. Problematisch sind allerdings Fehler der Prädiktion, die unnatürliche prosodische Realisierungen oder Aussprachen vorschreiben.

Eine weitere Methode der Korpusdefinition ist die Generierung von Sätzen, die eine hohe Dichte der gewünschten Einheiten aufweisen. Dadurch lässt sich die Abdeckung zwar optimieren, allerdings besteht die Gefahr, dass die resultierenden Phrasen- und Satzstrukturen unnatürlich bis hin zur semantischen Absurdität sind oder zumindest

eine geringe Idiomatik aufweisen. Insbesondere die domänenbeschränkte Synthese baut aber darauf, dass bestimmte lexikalische Einheiten und Folgen oder Phrasenstrukturen im Korpus enthalten sind, um das Ziel der annähernden Natürlichkeit zu erreichen. Natürlich lässt sich auch diese Anforderung in den Generierungsprozess einbauen, verlässlicher ist aber die Auswahl aus einem Textkorpus der sprachlichen Domäne. Black und Lenzo (2001) stellen ein Verfahren vor, das mit Hilfe des Clunits-Algorithmus eine Textvorlage für ein solches Korpus auf der Grundlage von Text aus der neuen Domäne und eines Synthesekorpus aus einer anderen geschlossenen oder der offenen Domäne erzeugt. Dabei werden die Einheiten des Sprachkorpus dem Clustering unterzogen. Im Anschluss wird jeder Satz aus dem Textkorpus synthetisiert und die Häufigkeit der Verwendung einzelner Cluster oder Einheiten aus dem alten Korpus in den Cluster-Entscheidungsbaum eingetragen. Somit erhält man eine Häufigkeitsverteilung, die als Grundlage für die Bewertung der Einheiten im Textkorpus dienen kann. Um zu verhindern, dass ausschließlich häufige Einheiten ausgewählt werden, bewertet man einmal selektierte Typen im Folgenden mit einer Punktzahl von 0. Black und Lenzo zeigen, dass dieses Verfahren Korpora erzeugt, die innerhalb der Zieldomäne qualitativ höherwertige synthetische Sprache liefern als in fremden Domänen.

Ein generelles Problem der datengetriebenen Synthese ist das Phänomen der sogenannten *large number of rare events*, LNRE (Möbius, 2003). Damit wird die Eigenschaft von Sprache beschrieben, dass sie eine hohe Zahl von Erscheinungen aufweist, die einzeln selten vorkommen, aber in ihrer Gesamtzahl so stark vetreten sind, dass mindestens eine von ihnen in fast jeder Äußerung auftaucht. Auf die Korpusdefinition übertragen könnten dies zum Beispiel Folgen seltener Laute sein, die auch in sehr großen Textgrundlagen nicht vollständig vorkommen. Jeder dieser Fälle mindert daher die Qualität der Synthese. Dieses Problem besteht auch für die geschlossenen Domänen. Der Grund ist, dass diese nur ein abstraktes Konzept darstellen, in der Realität hingegen das Vokabular und die Syntax in allen Anwendungsszenarien von Sprache jedoch offen sind, nicht zuletzt durch die große Zahl verschiedener Eigennamen (vgl. auch Kapitel 6 und Möbius, 2003). Im Gegensatz zum *phrase-slot-filling* kann es daher für die ,,echte" Sprachsynthese, die einen bestimmten Anwendungsfall und nicht nur eine streng begrenzte Kombination von Äußerungen abdecken soll, eine Geschlossenheit nicht geben. (Bozkurt et al., 2003) erzielen eine verbesserte Abdeckung auf derselben Textgrundlage, indem sie den Bewertungsalgorithmus so modifizieren, dass möglichst unterschiedliche Sätze ausgewählt werden. Sehr seltene Erscheinungen bleiben vermutlich jedoch problematisch, weil auch die datengetriebenen Verfahren zur phonetischen Annotation der Textkorpora von derselben *data sparsity* betroffen sind. Ein trainingsbasiertes Transkriptionsverfahren wird oft genau für seltene Fälle fehlerhafte Ausgaben generieren, so dass Unstimmigkeiten

zwischen der erwünschten bzw. „korrekten" Aussprache und der Annotation entstehen. Aussagen über die tatsächliche Abdeckung können dann erst bei und nach Aufnahme des Sprachkorpus gemacht werden.

Unabhängig von der Methode der Erstellung von Korpora wird im Anschluss in einigen Fällen ein Pruning des annotierten akustischen Signalkorpus durchgeführt, bei dem Instanzen ähnlicher Einheiten aus dem Korpus entfernt werden. Dies war z. B. bei den ATR-Synthesesystemen ν-Talk und CHATR eine Strategie zur Datenreduktion. Da die tatsächliche Variabilität der phonetischen Realisierung größer ist als die nach den Ähnlichkeitsmaßen messbaren Unterschiede und die in Abschnitt 2.5.2 diskutierten Probleme der mangelnden Korrelation dieser Maße mit der menschlichen Sprachwahrnehmung existieren, ist mit dem Pruning immer auch eine mehr oder weniger ausgeprägte Qualitätsreduktion verbunden.

Untrennbar mit der Korpusdefinition verbunden ist die Frage, welche Basiseinheiten für die Synthese verwendet werden. Während bei den ersten ATR-Versuchen Laute dynamisch zu Einheitenketten verschiedener Längen zusammengesetzt werden, sind es bei den Nachfolgern vorberechnete *non-uniform units*. Dies erschwert allerdings die Abschätzung der Verteilung und die Konstruktion von Synthesekorpora. Mit der Hinzunahme der Prosodie in CHATR ergibt sich außerdem das Problem der Beschreibung solcher nicht-uniformen Einheiten, indem z.B. Aussagen über die Grundfrequenz sowohl bei sehr langen als auch bei lautgroßen Einheiten über den Mittelwert dieses Parameters getroffen werden. Für die langen Einheiten, die möglicherweise mehrere Gipfel und Täler der Intonationskontur enthalten, ist eine solche Annotation viel weniger sinnvoll als für die kurzen. Auch für die Messung von akustischen Distanzen im Clunits-Verfahren eignen sich solche Beschreibungen nicht, weshalb dort auf Phone zurückgegriffen wird. Die nicht-uniforme Auswahl in der Verbmobil-Synthese ist von dem Problem weniger stark betroffen, da sie mit der Einheit Silbe genau den Wirkungsbereich des Intonationsparameters beinhaltet und F0-Mittelwerte damit eine größere Aussagekraft behalten. Problematischer sind auf Wortebene Komposita mit ggf. mehreren nebenbetonten Silben.

Die Qualitätsschwankungen zwischen der Auswahl langer Ketten und einzelner Phone sind für Conkie (1999) der Grund, sich in der AT&T-Synthese für uniforme Einheiten zu entscheiden. Dabei besinnen sie sich auf die traditionelle konkatenative Synthese zurück und verwenden die neu geschaffene Einheit Halbphon (*half-phone*)[7], aus der sich sowohl die althergebrachten Diphone als auch Phone zusammensetzen lassen. Letztere kommen nur zum Einsatz, wenn für einen Lautübergang kein Diphon im Korpus existiert. Wie schon die Entwickler des CHATR-Systems verlassen sich Conkie und Isard (1996) bei der Konkatenation nicht allein auf die Segmentation

[7]Zeitgleich entwickelten (Balestri et al., 1999) die äquivalenten *demi-phones*.

des Korpus, sondern suchen für zwei zu verbindende Einheiten anhand Cepstrum-basierter Maße die Überlappungsfläche mit der größten Ähnlichkeit, um dort zu verketten. Diese Technik nennen sie *optimal coupling* (vgl. auch Kraft, 1997).

Wohl aufgrund der einfacheren Handhabung und stabileren, wenn auch nicht notwendigerweise höheren Qualität scheinen auch die meisten neueren Systeme Diphone oder Halbphone, z. T. mit Fallback auf die Phonebene zu verwenden (Fraser und King, 2007). Grůber et al. (2007) vergleichen für das Tschechische verschiedene uniforme Einheiten und attestieren den fast gleichwertig performanten Halbphonen, Diphonen und Triphonen eine bessere Eignung gegenüber Phonen und Silben. Wegen der höheren Berechnungskomplexität beim Einsatz von Halbphonen empfehlen sie die Verwendung von Diphon- oder Triphon-Einheiten. Einen gemischten Einsatz dieser Einheitentypen testen sie allerdings nicht. Es verwundert kaum, dass die Silbeneinheiten im alleinigen Einsatz gegenüber den kleineren Einheiten schlechter abschneiden, da bei gleicher Korpusgrundlage natürlich viel weniger Kontextvarianten für längere Units existieren. Hier wäre es interessanter gewesen, einen Multilevel-Ansatz, wie er BOSS/Verbmobil zugrundeliegt, mit der einstufigen Auswahl zu vergleichen, da dort im Fall einer ungeeigneten phonetischen Umgebung auf kleinere Einheiten zurückgegriffen werden kann.

Das Thema Synthesebausteine wird im nächsten Kapitel im Verlauf der Schilderung eigener Arbeiten zur Definition von Einheiten weiter vertieft. Der Schwerpunkt liegt dabei, anders als in diesem Kapitel, auf der Relation von Symbol und Signal in einer phonetisch-phonologischen Betrachtungsweise des Problems, die gleichzeitig dennoch der Grundidee der Unit Selection verhaftet ist, explizite durch implizite phonetische Modellierung zu ersetzen.

3. Neudefinition von Syntheseeinheiten

„Jetzt wächst zusammen, was zusammen gehört.“ *Willy Brandt*

3.1. Motivation

Insbesondere bei der Entwicklung von Unit-Selection-Systemen scheint im Allgemeinen wenig Gewicht auf das Verhältnis zwischen dem Satz von Symbolen, die für die Transkription verwendet werden, und den Syntheseeinheiten gelegt zu werden. Die Forschung konzentriert sich häufig auf die Verbesserung spektraler Abstandsmaße, auf optimale Vorauswahl von Einheiten, Optimierung der Kostenfunktionen und die Abdeckung des Korpus. Alle diese Faktoren sind für eine qualitativ hochwertige Synthese natürlich ungemein wichtig. Damit verbunden ist aber leider die Problematik, dass die Verbesserung und Entwicklung von Unit-Selection-Synthese als rein informationstechnische Aufgabe betrachtet und behandelt wird. Dabei beklagen Sagisaka et al. noch 1992:

> ...problems seem to result from the development history of text-to-speech systems. In text-to-speech systems, efforts have mainly been put on the scientific investigation of control principles and fundamental models and not so much attention has been paid to the technological aspects such as algorithmic modeling or system optimization.

Hingegen scheint sich die Forschung heute fast ausschließlich mit den letztgenannten Aspekten der Sprachsynthese zu befassen – auf Kosten der phonetischen Modellierung der Sprachproduktion. Taylor und Black (1999) weisen denn auch darauf hin, dass der Anteil der phonetischen Beiträge zum Thema im Laufe der Jahre stark gesunken ist. Als Grund führen sie an, dass die explizite Modellierung noch nicht detailliert genug ist, um natürliche Sprache zu erzeugen und daher der datengesteuerte Unit-Selection-Ansatz, der vermeintlich größtenteils ohne diese auskommt, so

erfolgreich ist. Sie plädieren daher dafür auch im segmentalen Bereich stärker zu abstrahieren und kanonische (dort ,,phonologische") statt enger Transkriptionen für die Einheitenauswahl zu verwenden. Dass dies aber keinen Widerspruch zur phonetischen Analyse darstellt und diese letztlich die Voraussetzung für die Verwendbarkeit dieses Ansatzes ist, soll in diesem Kapitel gezeigt werden.

Nach Taylor und Black (1999) wurde die Einheitenauswahl aus Korpora zum Zeitpunkt der Veröffentlichung über eine enge Transkription der Zieläußerung vorgenommen. Ob dies verallgemeinert werden kann ist schwer zu überprüfen, denn oft existieren für die verschiedenen Systeme keine Konventionen für die Transkription (oder werden zumindest nicht veröffentlicht), die über eine bloße Aussage wie ,,Wir verwenden SAMPA[1] mit den folgenden Abweichungen ..." hinausgehen. Dadurch wird aber letztlich nur ein Symbolinventar beschrieben. Fragen, die die phonetische Qualität bestimmter strittiger Fälle betreffen, die gesamte Phonotaktik inkl. Silbenschnitt und das Verhältnis zwischen dem Standard und der Transkription einerseits und dem Standard und der Realisierung durch den Sprecher andererseits, werden ausgeklammert. Damit ist aber der Boden bereitet für Inkonsistenzen zwischen der Transkription, wie sie vom Synthesesystem zur Laufzeit erzeugt oder aus aus einem Lexikon ausgelesen wird, und der Annotation des verwendeten Korpus.

Ein weiterer Effekt der durch das Unit-Selection-Paradigma stark verbesserten Natürlichkeit der Synthese ist offenbar, dass die Forschungsergebnisse aus der traditionellen Forschung zu konkatenativen Verfahren, die in Erweiterung des Diphonansatzes zum Entwurf von Mischinventaren aus Halbsilben und weiteren Einheiten (Portele, 1996) geführt haben, für überholt gehalten werden: ,,If your models are right, it doesn't matter which units you use" (Nick Campbell, pers. Komm.). Diese Aussage soll im Folgenden einer kritischen Betrachtung unterzogen werden.

Beide beschriebenen Tendenzen in der Forschung zur konkatenativen Sprachsynthese führen unmittelbar zur Frage der Einheitendefinition und des zu verwendenden Transkriptionsformats, die im nächsten Abschnitt behandelt werden soll.

3.2. Phoxsy: Multiphon-Einheiten für die Einheitenauswahl aus großen Korpora

Unit-Selection-Synthese soll sprachliche Variation implizit liefern, daher wird, mit Ausnahme von Dauer und Phrasierung, auf fehlerträchtige prosodische Modellierung und perzeptiv störende Signalmanipulation oft weitestgehend oder ganz verzichtet.

[1]SAMPA ist ein maschinenlesbares phonetisches Alphabet auf der Basis von 7-Bit-Zeichen (Wells, 1997, 2005).

Die phonologische und phonetische Variabilität auf der segmentellen Ebene wird dabei jedoch nur scheinbar abgedeckt. Zwar werden i. d. R. Silben, Phone oder andere Syntheseeinheiten nach Möglichkeit aus einem phonetischen Kontext ausgewählt, der dem der Zieläußerung entspricht, oft ist jedoch weder gewährleistet, dass die Zielvorgabe eine sinnvolle phonologische Tiefenstruktur aufweist, noch dass sie eine kontextadäquate Oberflächenstruktur beschreibt. Anders ausgedrückt werden also keine Maßnahmen ergriffen um sicherzustellen, dass die von der Synthese generierte Transkription zur idiosynkratischen Sollaussprache des Sprechers passen oder dass eine Verbindung zwischen dieser Transkription und den tatsächlich realisierten Aussprachen eines Wortes an verschiedenen Stellen des Korpus hergestellt werden kann. Probleme, die sich aus dem ersten Punkt ergeben, können dadurch vermieden werden, dass bei der Aufnahme eines Korpus darauf geachtet wird, dass der Sprecher sich entweder strikt nach der Vorgabe richtet oder das Lexikon der Aussprache des Sprechers angepasst wird. Der zweite Punkt stellt eine größere Herausforderung dar. So kann z. B. die Auswahl geeigneter Segmentketten aus dem Korpus dadurch verhindert werden, dass die Transkriptionsalgorithmen oder -lexika der Synthese eine kanonische Einzelwortaussprache vorgeben, die Annotation des Signalinventars jedoch die tatsächliche Realisierung abbildet — nicht nur im Fall von Elisionen ist dies ein schwerwiegendes Problem. Ein Lösungsansatz, von dem Taylor und Black 1999 offenbar noch annehmen, dass er in den meisten Systemen existiert, wäre, die Variation der segmentalphonetischen Realisierung durch manuell erstellte oder statistisch berechnete Transformationsregeln auf Satzebene zu modellieren. Ein solcher Ansatz für Äußerungen von deutschen Funktionswörtern wurde von Breuer (2000) präsentiert. Auch Bennett und Black (2003) modellieren die segmentale Variabilität und verwenden akustische Modelle aus der Spracherkennung, um Varianten vorherzusagen. In dieser Publikation wird dann auch der Eindruck geteilt, dass die Betrachtung der segmentalen Variation keine Tradition in der Unit-Selection-Forschung hat:

> ,,Within-speaker pronunciation variation is a well-known phenomenon; however, attempting to capture and predict a speaker's choice of pronunciations has been mostly overlooked in the field of speech synthesis."

Im Idealfall kann mit den Methoden der Modellierung sicherlich ein Match zwischen einer Quell- und Zieläußerung hergestellt werden. Es ergeben sich dabei jedoch folgende theoretische und praktische Probleme. Zum einen suggeriert die satzphonetische Vorhersage einer bestimmten Segmentfolge, dass ein Laut a aus einem Kontext $x\_(y)\_a\_z$ mit dem Laut a aus dem Kontext x_a_z zu vergleichen ist, wobei die Klammerung für eine Elision des Lautes y steht. Beispielsweise besteht jedoch für eine Reduktionsform [ʀaːm] des Wortes ,,Raben" die Möglichkeit, dass das assimilierte [m] durch seinen elidierten Vorgänger [b] einen wahrnehmbaren Timingunterschied

im gestischen Ablauf beinhaltet, der ihn von anderen [m] im Kontext [aː] unterscheidet[2]. Um dies berücksichtigen zu können, wäre eine sehr enge Transkription des Korpus notwendig, die für die großen Unit-Selection-Korpora nur mit erheblichem Zeitaufwand erstellt werden könnte. Ein anderes Problem solcher Modelle ist, dass sie eng an die (ggf. schwach modellierte) prosodische Prädiktion geknüpft sein müssen, um Wechselwirkungen zu berücksichtigen und die segmentalphonetische Variation der passenden prosodischen Variation zuzuordnen. Dass dies zuverlässig möglich ist, darf bezweifelt werden, zumal für die Realisierung auch pragmatische und semantische Aspekte eine Rolle spielen, die von der Vorverarbeitung eines TTS-Systems nur in Ansätzen modelliert werden können.

> „... as unit selection techniques use a narrow phonetic transcription as input[3], phenomena such as vowel reduction and assimilation have to be modelled before unit selection operates and the modules which govern this may make errors in the same way." (Taylor und Black, 1999)

Zudem würde man einen der Hauptvorteile des Unit-Selection-Ansatzes verschenken: Konsequenter wäre es, ein Verfahren zu wählen, das die dem Paradigma innewohnende Mächtigkeit der impliziten Abdeckung sprachlicher Variation auch auf der Segmentebene ausschöpft. Zwar haben Taylor und Black (1999) Recht, wenn sie dies durch Verwendung einer kanonischen Transkription versuchen. Ihr Ansatz geht jedoch nur auf, wenn die Anzahl der kanonisch vorgegebenen Segmente und der zu beschreibenden Korpuseinheiten übereinstimmt. Hier setzt das in diesem Kapitel beschriebene *Phoxsy*-Konzept (*Pho*ne e*x*tensions for *sy*nthesis) an. Es definiert Phon- und Multiphoneinheiten, die u. a. ermöglichen, bestimmte Realisierungen von Morphemen und Phonemfolgen mit identischen Symbolen zu belegen ([ən]/[n̩]), unabhängig von den tatsächlich verwendeten freien allomorphischen und allophonischen Varianten. Diese abstrahierende Beschreibung der kanonischen Repräsentation und der Annotation soll bewirken, dass Unterschiede in der symbolischen Darstellung nivelliert werden, um eine möglichst große Nutzbarkeit des Korpus und gleichzeitig eine hohe Qualität zu erreichen. Dies funktioniert für die berücksichtigten, von Variation häufig betroffenen Phonemfolgen inter- und intrasubjektiv, so dass auch keine Anpassung für verschiedene Synthesekorpus-Sprecher notwendig ist, sofern diese hinreichend standardnah sprechen.

Die Basis der Phoxsy-Einheiten bildet die Spezifikation boss-sampa_DE, die aus einem ASCII-basierten phonetischen Alphabet und Anweisungen für die Transkription besteht. Das Alphabet bedient sich der in SAMPA-D, X-SAMPA (Wells, 1997,

[2]Dieses Beispiel repräsentiert allerdings ein Extrem der Reduktion und Assimilation, das in der Sprachsynthese aus Gründen der Verstehbarkeit und Verständlichkeit i. A. unerwünscht wäre.
[3]Hiermit ist die Korpusannotation und nicht das Ergebnis der automatischen Transkription als Zielvorgabe zur Laufzeit gemeint.

2005) und SAMPA-D-VMLex (Gibbon, 1995) vorgeschlagenen Symbole zur Transkription des Deutschen und der in der deutschen Synthese verwendeten Xenophone (fremdsprachlichen Laute). Eine vollständige Übersicht findet sich in Anhang A. Die Vorschriften für die Transkription und Annotation geben eine breite phonetische Beschreibung der Einzelwortaussprache für alle Wörter vor. Dabei werden im Unterschied zum Duden gespannte Kurzvokale, auch in Klassizismen, immer als Langvokale beschrieben, weil die Unterscheidbarkeit bereits redundant über die Nicht-Akzentuierung dieser Einheiten gegeben ist. Rein phonologisch unterscheidbare aber homophone Einheiten werden nach Möglichkeit zusammengefasst ([i]/[j]). Alle Wörter werden nach dem *Maximum Onset Principle* annotiert, so dass die Silbengrenze immer vor dem phonotaktisch längsten erlaubten Anlaut der Nachfolgersilbe liegt. Eingeschränkt wird diese Ausdehnung jedoch durch Morphemgrenzen, die nur im phonotaktisch minimal erforderlichen Rahmen überschritten werden sollen. Jedes Wort erhält minimal einen Wortakzent[4] und optional einen Sekundärakzent. Die Symbole für die Silbengrenzen und Akzente werden mit den weiteren im boss-sampa-Standard enthaltenen nicht-segmentalen Annotations- und Transkriptionszeichen in Abschnitt 4.1.1 vorgestellt.

Ein weiteres Ziel der Phoxsy-Definition ist, die Segmentierung des Signals an solchen Stellen zu vermeiden, an denen die präferierte Grenze zwischen zwei Segmenten schwer zu bestimmen und ein konsistentes Setzen des Schnittpunktes für verschiedene oder denselben Annotierer nicht trivial ist, bzw. an Stellen, an denen die Fehlerrate der automatischen Segmentierung in die Höhe schnellt. Dies betrifft insbesondere Übergänge von Lauten wie Approximanten und Liquiden im Silbenanlaut, die kaum stationäre Anteile besitzen und stark kontextabhängig geprägt sind, zu Vokalen. Die Zusammenfassung dieser Lautgruppen mit folgenden Vokalen zu Multiphon-Einheiten beruht auf der Annahme, dass die zugrundeliegenden Einheiten ohnehin i. d. R. nur in der im Quellsignal gegebenen Abfolge sinnvoll konkateniert werden können, da die Positionierung der Lautgrenzen und die Realisierung der Anlautkonsonanten hochspezifisch für eine konkrete Folge sind. Phoxsy folgt damit im Grunde einer ähnlichen Motivation zur Einheitendefinition wie die klassische konkatenative Synthese von kleinen Inventaren mit einer 1:1-Relation zwischen Klassen und Instanzen von Einheiten, ist aber an die speziellen Bedürfnisse der Unit-Selection-Synthese angepasst, insbesondere, wie sie durch BOSS realisiert ist. Anders als bei früher verwendeten Mischinventaren kann die Anzahl der Einheitentypen jedoch durch die Möglichkeit der kontextbezogenen Auswahl aus mehreren Instanzen deutlich reduziert werden. In der Phoxsy-Spezifikation werden daher nur noch solche Fälle abgedeckt, die hochvariabel und gleichzeitig nicht immer sinnvoll

[4]Es ist geplant, dies für einsilbige Funktionswörter zu ändern, damit deren mit hoher Wahrscheinlichkeit reduzierte Segmente nicht als akzentuierte Einheiten von Inhaltswörtern verwendet werden.

segmentierbar sind. Ein positiver Nebeneffekt der Multiphon-Spezifikation ist, dass die Kombinatorik in der Einheitenauswahl durch paradigmatische (weniger Einheiteninstanzen pro Typ) als auch syntagmatische Reduktion (weniger Einheiten pro Äußerung) ein geringeres Problem darstellt als in rein phon-basierten Ansätzen. Dies führt zu Laufzeitvorteilen für die Synthese.

3.2.1. Beispiele für zusammengesetzte Lautfolgen

Im Folgenden werden einige Beispiele für die in Phoxsy zu einer Einheit zusammengefassten Lautfolgen anhand von Ausschnitten aus dem Unit-Selection-Korpus 2 des BITS-Projektes (Ellbogen et al., 2004) und dem Verbmobil-Korpus gegeben. Dateien die aus dem BITS-Korpus stammen sind durch mit `US1002` beginnende Dateinamen in der Bildunterschrift markiert; die Namen von Verbmobil-Dateien beginnen dagegen mit `a0`. Die Annotationen werden so wiedergegeben, wie sie in den Korpora vorkommen. Die Verbmobil-Daten wurden, wie die BITS-Signale, maschinell segmentiert, allerdings im Gegensatz zu diesen nicht manuell nachkorrigiert.

Folgen von [j] und Vokalen sind häufig schwer zu separieren, weil sie sich phonetisch wie ein Diphthong verhalten und damit im Übergang vom Halbvokal zum Vokal keine stationäre Phase beinhalten[5]. Abbildung 3.2.1 zeigt dies beispielhaft für die Folge [ja]. Auditiv ist hier keine vollständig zufriedenstellende Grenze zu finden und auch nach visuellen spektralen Kriterien gäbe es einige Freiheitsgrade bei der Grenzsetzung.

Selbst wenn eine maschinelle Segmentierung in verschiedenen Instanzen der Folge immer einen vergleichbaren Schnittpunkt nach Formantlage finden könnte, wären die Dauerverhältnisse und Transitionsgeschwindigkeiten der separaten Segmente vermutlich unterschiedlich genug, um in anderen Zusammensetzungen perzeptiv auffällig zu werden. Eine Technik wie das *optimal coupling* (s. Abschnitt 2.5.3) oder ähnliche Maßnahmen zur Ermittlung eines optimalen Konkatenationspunktes zwischen zwei Segmenten könnten spektrale Störungen am Übergang zwar mindern, dies aber womöglich noch stärker auf Kosten natürlicher Dauerverhältnisse der dem [j] und dem Vokal zugeordneten Signalabschnitte. Die Frage, ob an diesen Stellen eine Segmentierung überhaupt notwendig und sinnvoll ist, scheint daher berechtigt. Schwieriger noch wird die Suche nach dem Schnittpunkt, ob maschinell oder durch den Menschen gesetzt, wenn der Folgevokal, wie in Abbildung 3.2.2, einen [j] ähnlichen Artikulationsort besitzt. Hier wird die auditive Grenzsetzung vollends willkürlich.

[5] bzw. diese erst spät im Wahrnehmungsbereich des Folgevokals beginnt.

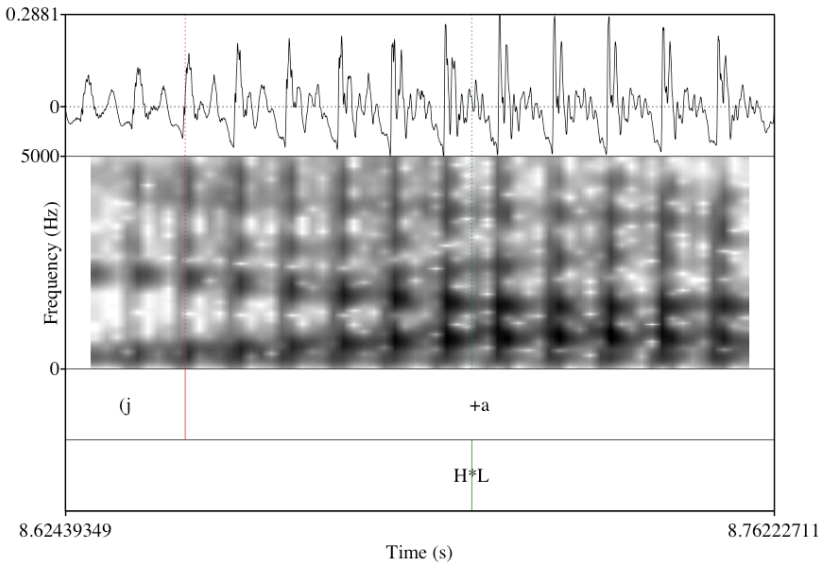

Abbildung 3.2.1.: [ja] in der Folge /n jan/ – US10020008

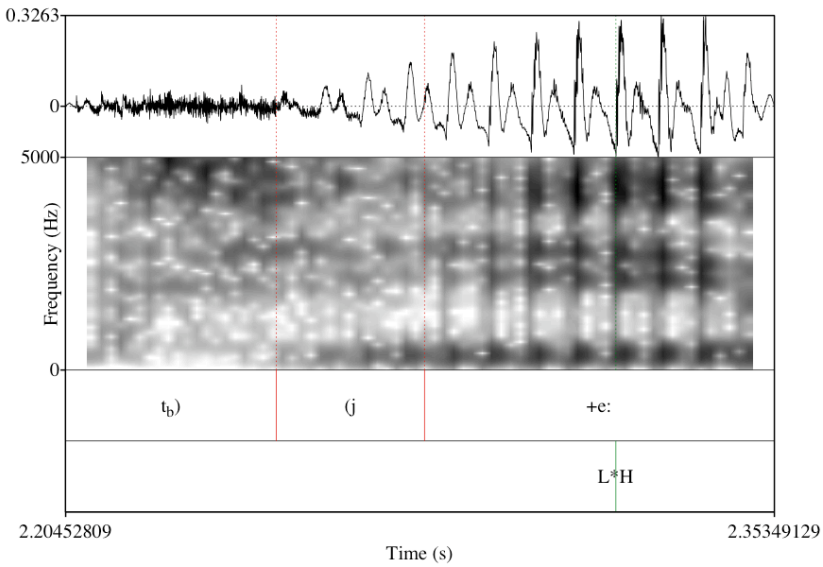

Abbildung 3.2.2.: [t.jeː] in der Folge /t.jeːm/ – US10020649

In Abbildung 3.2.3 ist zusätzlich noch der Vorgängervokal fast identisch mit dem Artikulationsort des Approximanten. Hier wäre im Prinzip eine Definition der gesamten vokalischen Folge als Einheit gerechtfertigt. Um die Kompatibilität zur hierarchischen Multi-Level-Auswahl von BOSS zu wahren, überschreiten Phoxsy-Einheiten jedoch prinzipiell keine (kanonischen) Silbengrenzen. Damit bleibt auch, anders als bei den in Abschnitt 2.1 dargestellten non-uniform units eine sinnvolle prosodische

Beschreibbarkeit der Einheiten erhalten. Eine Aussage zu den Grenzen der Koartikulationseinflüsse stellt diese Entscheidung nicht dar.

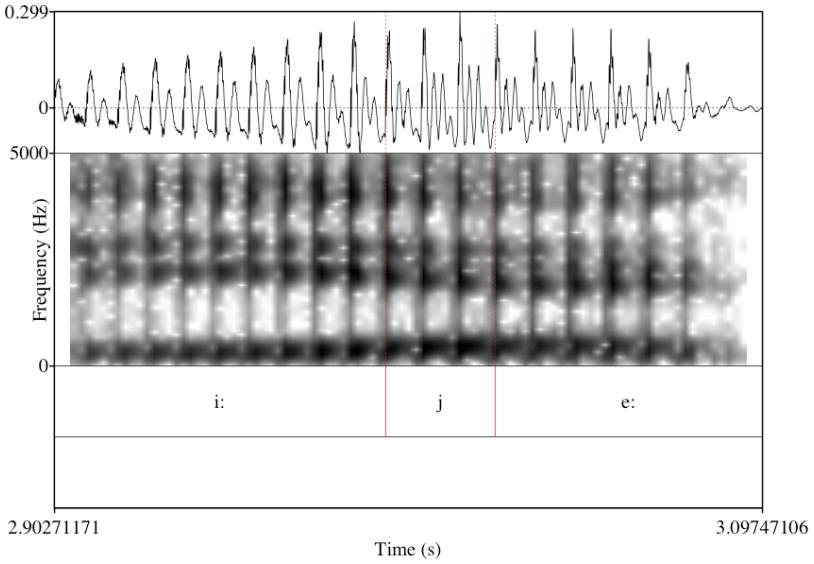

Abbildung 3.2.3.: [iːjeː] in der Folge /iːjeːk/ – US10020175

Als behauchte Version des Folgevokals (ggf. mit Transitionen von einem direkten oder indirekten Vorgängerlaut) passt die **Realisierung von /h/** ideal zu genau der Instanz des Vokals, mit der es im Korpus auftaucht. Potenziell wäre die Kombination sicher auch mit anderen Vokalinstanzen möglich. Da die zeitliche Ausdehnung der Behauchung sich allerdings über den ganzen Vokal erstrecken kann, ohne dass ein separierbares [h] vorausgeht, kann die phonologische Folge /h/+Vokal in dieser Realisierungsform als Vokal mit behauchter Stimmqualität betrachtet werden. Dann ist die zeitliche Segmentierung jedoch arbiträr und kann bei der Konkatenation einzelner [h]-Varianten mit Vokalen zu Unstimmigkeiten der Dauerverhältnisse führen, wie bereits bei der Darstellung der [j]-Varianten ausgeführt. Solche Fälle illustriert die Realisierung von /hoːr/ als [o̤ːɐ̯] in Abbildung 3.2.4. Die Realisierung ist jedoch selbst in segmental ähnlichen Kontexten einer starken Variabilität unterworfen. Abbildung 3.2.5 zeigt eine gut vom Folgevokal trennbare stimmhafte Variante [ɦi] mit dem gleichen linken Kontext [n]. In Abbildung 3.2.6 wird dagegen, bedingt durch den stimmlosen Frikativ, eine ebenfalls stimmlose Variante [h] realisiert. Angesichts dieser Variationsbreite, innerhalb der sich /h/ sowohl segmental als auch suprasegmental manifestieren kann, ist der Verzicht auf einen Schnitt zwischen /h/-Allophonen und Folgevokalen sicherlich sinnvoll, um Segmentierungsunstimmigkeiten zu vermeiden, die durch *forced alignment* einer kanonischen Zielsequenz oder Unsicherheiten bei der manuellen Segmentierung entstehen.

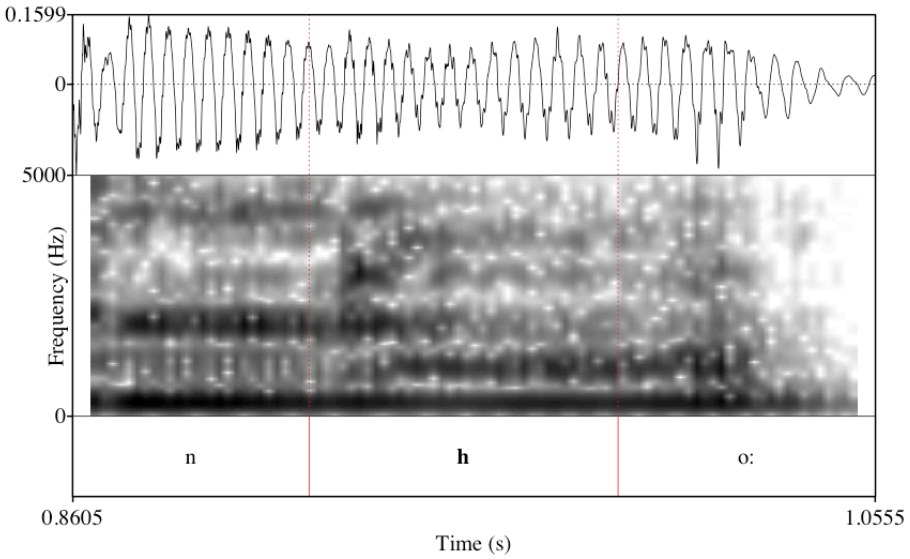

Abbildung 3.2.4.: [o̞ː] in der Folge /n hoːt/ – a00026

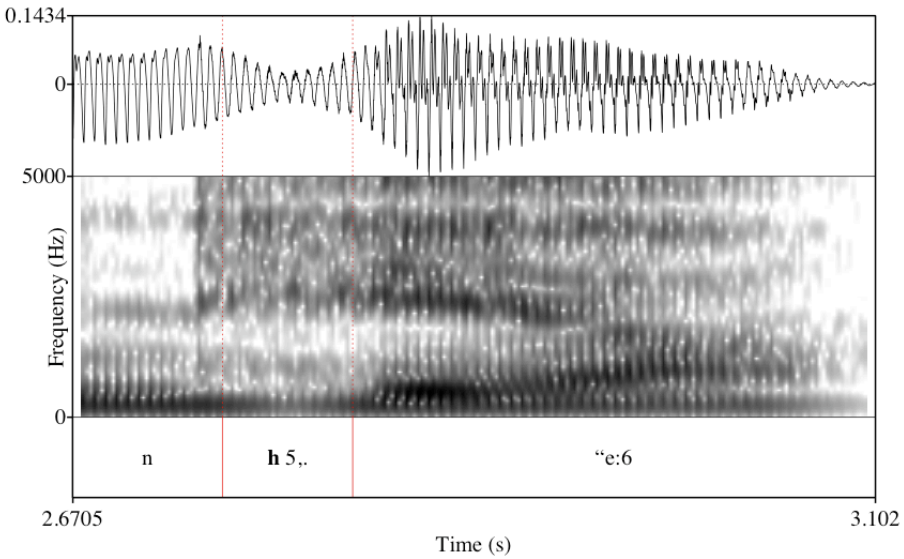

Abbildung 3.2.5.: [ɦ] in der Folge /n heːr‖/ – a00755

Für den Glottalverschluss /ʔ/ gilt ähnliches wie für /h/, d.h. die Realisierung findet oftmals nicht in Form eines eigenständigen Segmentes [ʔ], sondern als Glottalisierung des gesamten Vokals statt. Wie dort tauchen dabei verschiedene Kombinationen von Segmenten und Suprasegmentalia auf, so dass für die Nicht-Segmentierung von /ʔ/-Vokalfolgen die gleiche Argumentation angewendet werden kann. Abbildung 3.2.7 zeigt die Realisierung des Glottalverschlusses als stark laryngalisierten

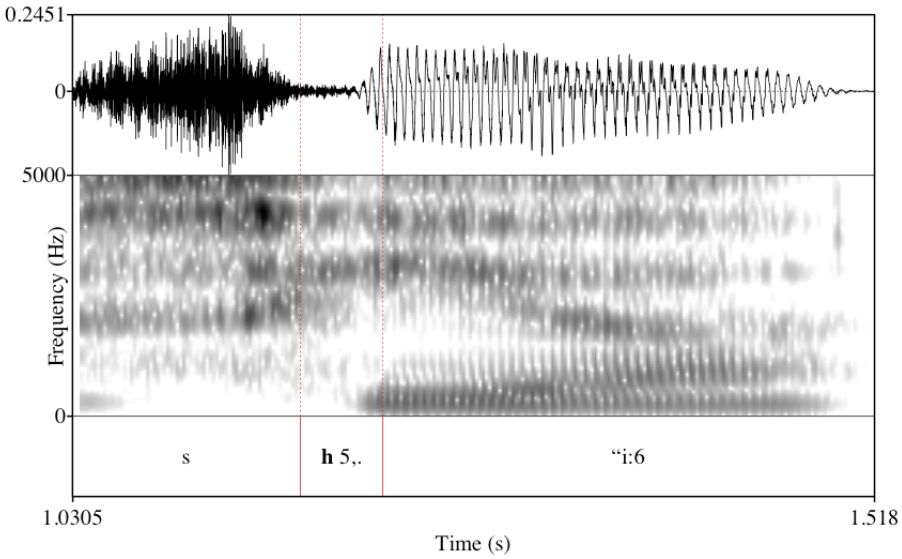

Abbildung 3.2.6.: [h] in der Folge /s hiːr‖/ – a00163

vorderen Anteil des Vokals [aː]. Dagegen führt die gleiche ,,intervokalische” Position in Abbildung 3.2.8 zu einer durchgängigen Glottalisierung und damit zur Nicht-Segmentierbarkeit. In Abbildung 3.2.9 sieht man wiederum einen Mehrfachverschluss mit anschließender suprasegmentaler Realisierung.

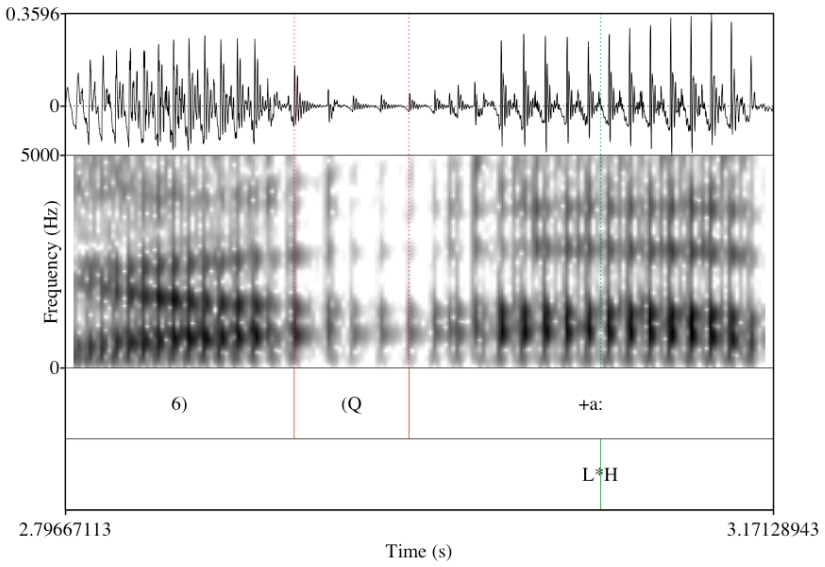

Abbildung 3.2.7.: [a̰ː] in der Folge /ər ʔaːˌb/ – US10020004

Auch das Phonem /l/ kennt viele Arten der Realisierung. Abbildung 3.2.10 zeigt

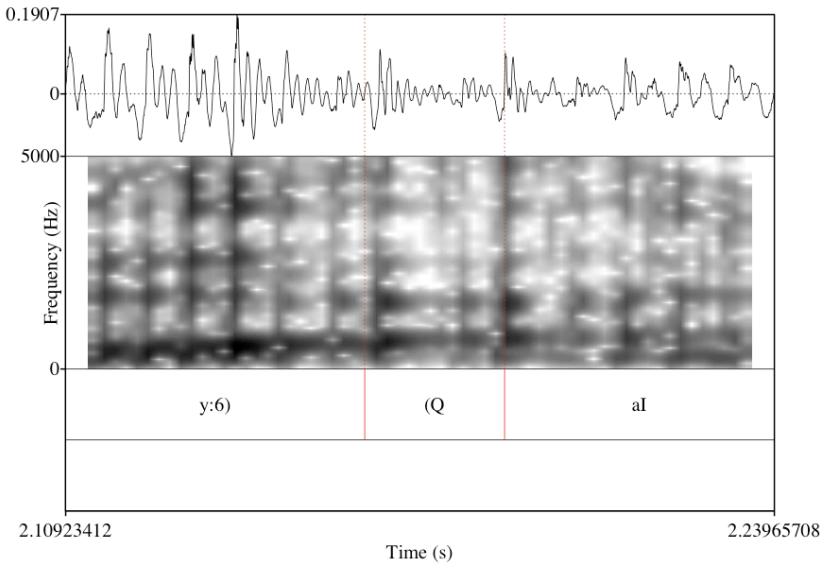

Abbildung 3.2.8.: [a͡ɪ] in der Folge /r ʔam/ – US10020088

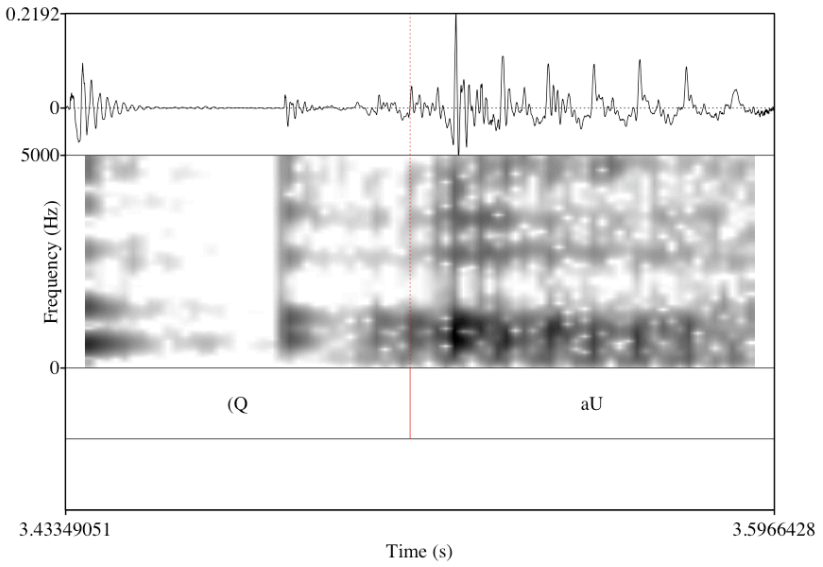

Abbildung 3.2.9.: [ʔa͡ʊ] in der Folge /t ʔaʊf/ – US10020004

eine relativ gut segmentierbare Instanz der Folge [lɛ], die allerdings aufgrund der langsamen Transition auch nicht ganz zufriedenstellend in perzeptiv disjunkte Einheiten segmentiert werden kann. Im gezeigten Fall wäre bspw. eine Grenzsetzung hinter der folgenden Grundperiode mit der gleichen Rechtfertigung möglich.

In Abbildung 3.2.11 manifestiert sich das /l/ in Form einer deutlich sicht- und

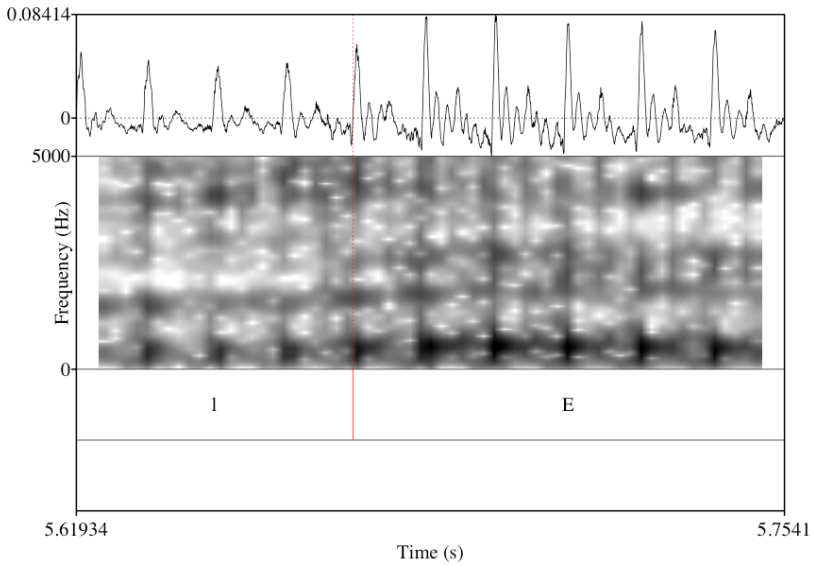

Abbildung 3.2.10.: [lɛ] in der Folge /ɔʏ.lɛt/ – US10020010

hörbaren lateralen Lösung während des vorangehenden [s]. Die als [l] und [ɛ] markierten Segmente erzeugen für sich genommen keinen deutlichen perzeptiven Eindruck einer Folge [lɛ]. An diesem Punkt ist fraglich, warum eine Segmentierung vorgenommen werden sollte, wenn dazu das mit [l] assoziierte Segment gerade einmal eine Grundperiode enthält. Auch wenn ein wichtiger akustischer Hinweis auf die [l]-Artikulation im vorangehenden Frikativ steckt, sollte der Beginn der Phoxsy-Einheit nicht bis zu diesem Punkt vorgezogen werden. Hier müssen die Mechanismen der Einheitenauswahl dafür Sorge tragen, dass das [s] nur in einen adäquaten Folgekontext eingebunden wird.

Abbildung 3.2.12 zeigt einen Abschnitt der äußerungsinitialen Silbe [plʊs], die aufgrund des starken Energieanstiegs eine sehr frühe Grenzsetzung aufweist. Eine optimale auditive Trennung setzt aber erst bei einer zwei bis drei Grundperioden später platzierten Grenze ein. Während das Energie-Kriterium hier für eine nachvollziehbare, wenn auch perzeptiv unbefriedigende Segmentierung sorgt, kann die Aufteilung in Abbildung 3.2.13 hingegen auf beiden Ebenen nicht überzeugen und muss als fehlerhaft bezeichnet werden. Hier ist also eine Inkonsistenz der Entscheidungen selbst in der automatischen Segmentierung festzustellen, die sich auf die Wiederverwertbarkeit der einzelnen Elemente in anderen, auch segmental identischen Kontexten negativ auswirken kann.

Zusätzlich zur kontextbedingten Variation verfügt **das Phonem /r/** im Anlaut zudem über eine Vielzahl von Allophonen verschiedener Lautklassen. Für Sprecher, die nicht über ein System von apikalen Vibrant- und reduzierten Schlag-Realisierungen

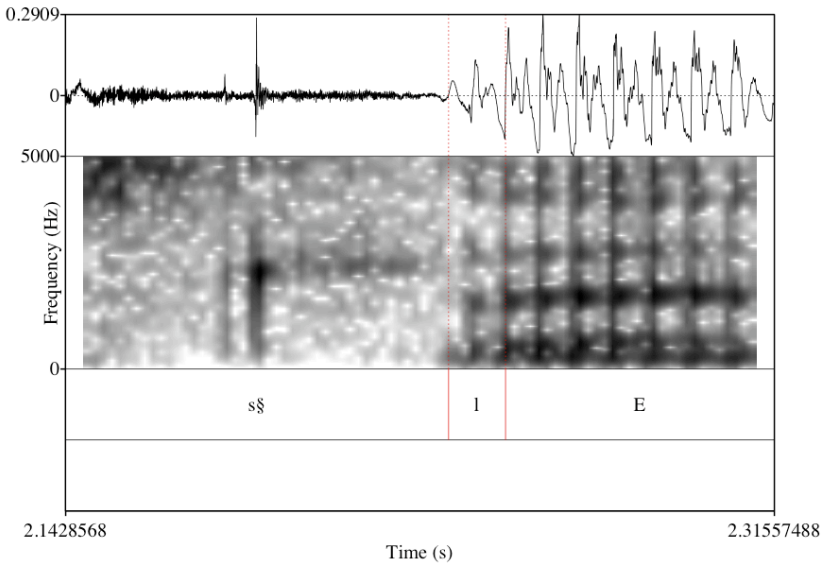

Abbildung 3.2.11.: [s.lɛ] in der Folge /s.lɛn – US10020344

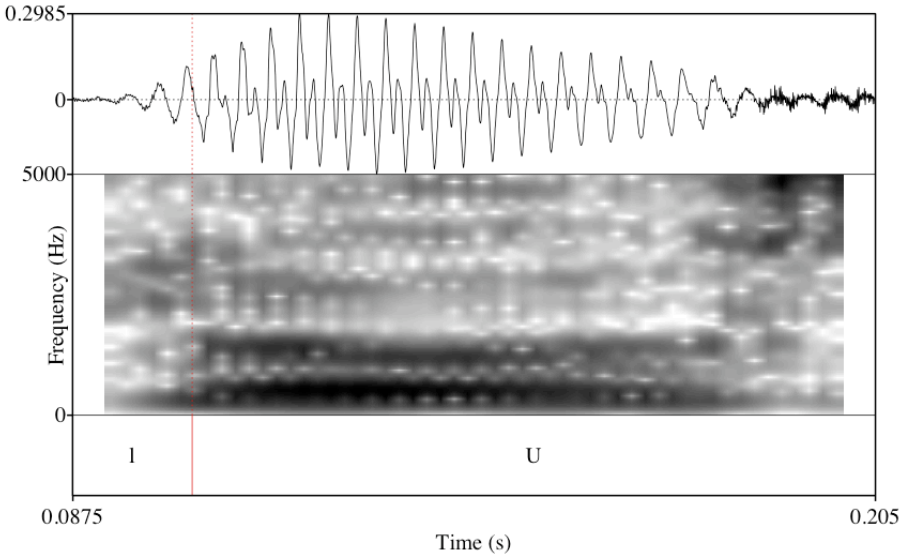

Abbildung 3.2.12.: [lʊ] in der Folge /plʊs/ – a00678

verfügen, gibt es die Möglichkeit der Realisierung als uvularen Vibranten oder dessen homorgane Reduktionsformen in den Artikulationsarten Frikativ und Approximant. Der Frikativ kann zudem stimmhaft oder lenis entstimmt artikuliert werden. Die Abbildungen 3.2.14, 3.2.15, 3.2.16 und 3.2.17 zeigen Beispiele für diese Varianten, die alle demselben Korpus entstammen. Das Beispiel für den Approximanten (3.2.17) demonstriert, dass Sprecher offenbar über große Freiheitsgrade verfügen und die

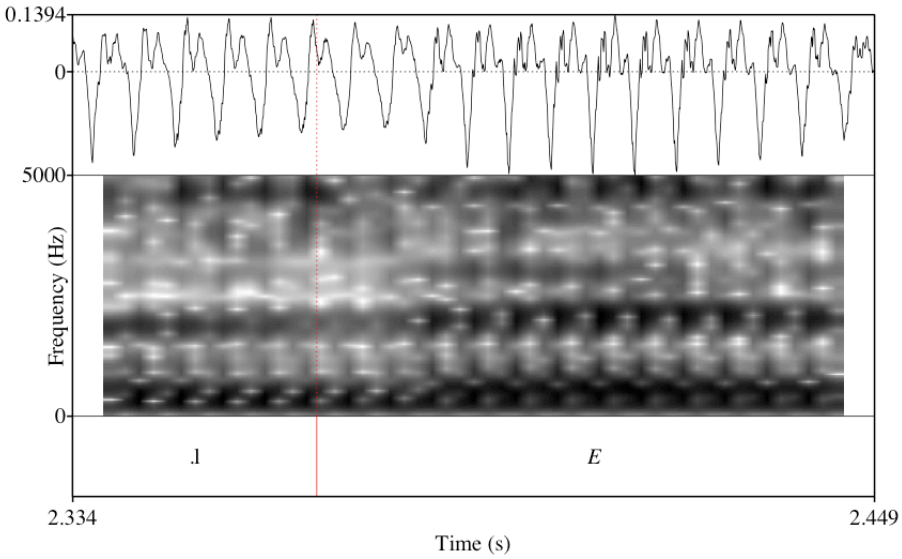

Abbildung 3.2.13.: [lɛ] in der Folge /a.len/ – a00313

Variabilität nicht allein aus dem jeweiligen Lautkontext erklärbar ist, denn in der Umgebung von [f] hätte man eher eine frikativische Realisierung erwarten können. Der Grund für die größere artikulatorische Sorgfalt liegt hier möglicherweise in der Position der Variante in einem akzentuierten Inhaltswort.

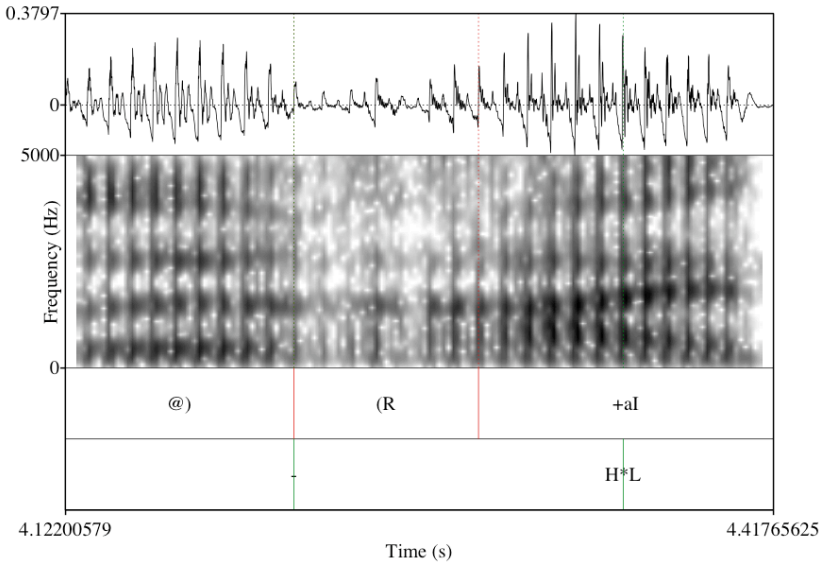

Abbildung 3.2.14.: [ʀ] in der Folge /ə raɪts/ – US10020080

Auch das phonologisch **postvokalische /r/** ist sehr variabel. Es kann mit dem Sil-

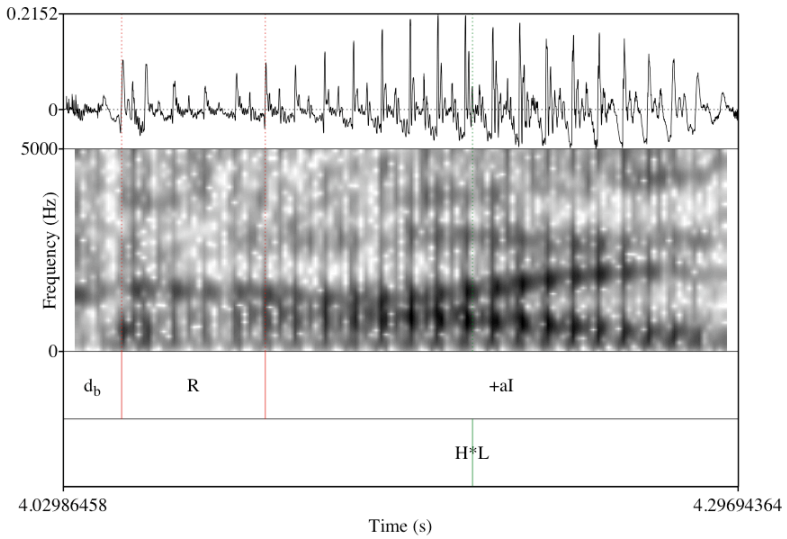

Abbildung 3.2.15.: [ʁ] in der Folge /draɪ.s/ – US100020009

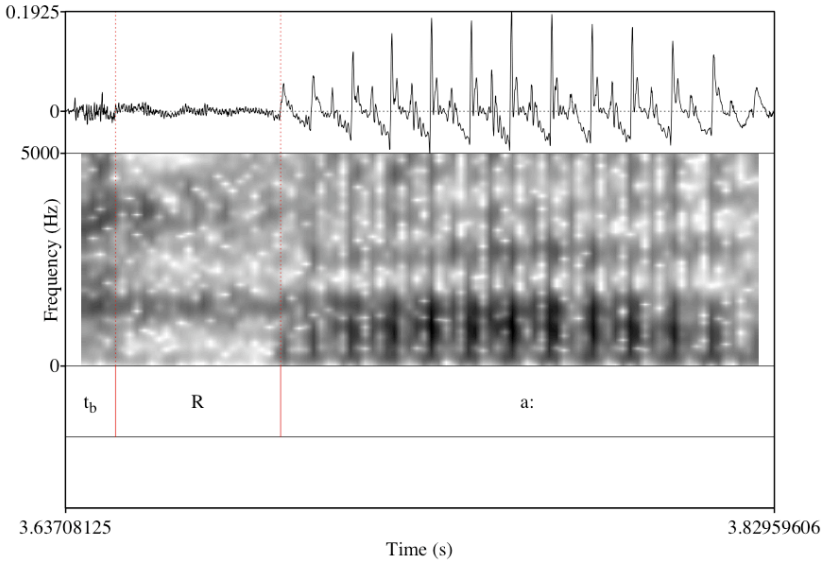

Abbildung 3.2.16.: [ʁ̥] in der Folge /traː.s/ – US10020009

benkern zu einem Diphthong verschmelzen und zusätzlich oder alternativ in einer der obengenannten Realisierungen als eigenes Segment auftauchen. Laut Duden-Aussprachewörterbuch (Mangold, 1990) ist der Standard für Kombinationen mit Kurzvokal-/r/ die Artikulation als Monophthong-Konsonant-Folge, während Langvokale zu Diphthongen werden. Dass in der Praxis bei zusammenhängender Sprache auch bei professionellen Sprechern alle Kombinationen möglich sind, zeigen die Ab-

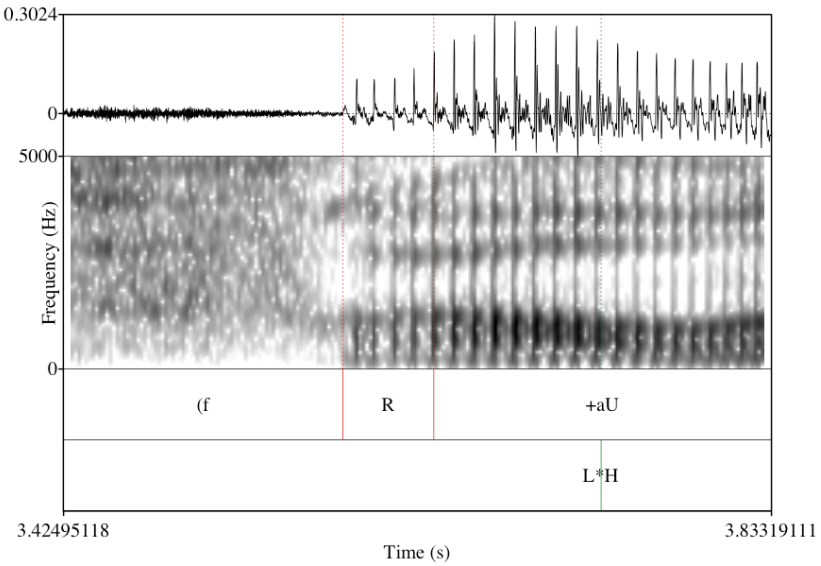

Abbildung 3.2.17.: [ɻ̩] in der Folge /fraʊ.ə/ – US10020022

bildungen 3.2.18, 3.2.19, 3.2.20 und 3.2.21. In der Konsequenz wird in der Phoxsy-Definition jegliche silbeninterne Vokal+/r/-Kombination als Diphthong etikettiert. In der Segmentierung fallen daher alle Abschnitte, die einer /r/-Realisierung in der Koda zugeordnet werden können, dem als Diphthong markierten Bereich zu.

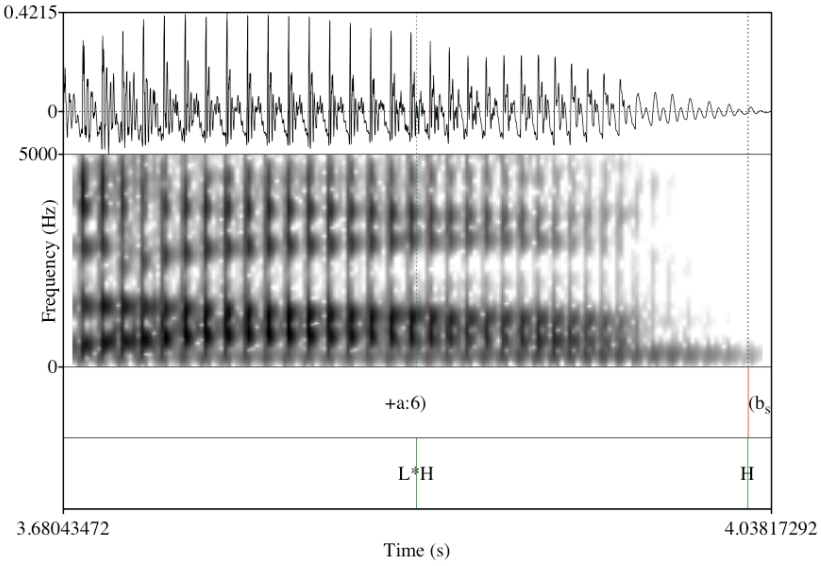

Abbildung 3.2.18.: [aːɐ̯] in der Folge /daːr b/ – US10020186

Phonologisch wird **das deutsche /v/** traditionell als Frikativ betrachtet. In der

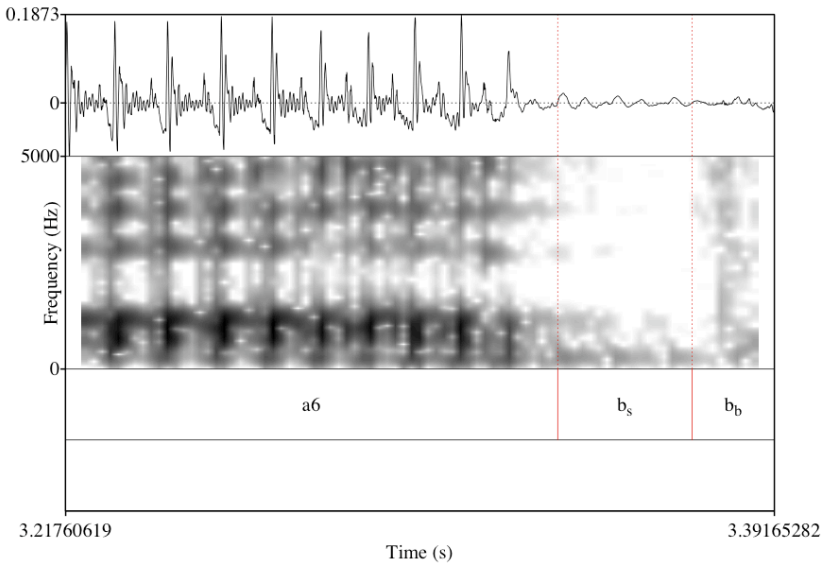

Abbildung 3.2.19.: [aɐ] in der Folge /ʔaːr.b/ – US10020058

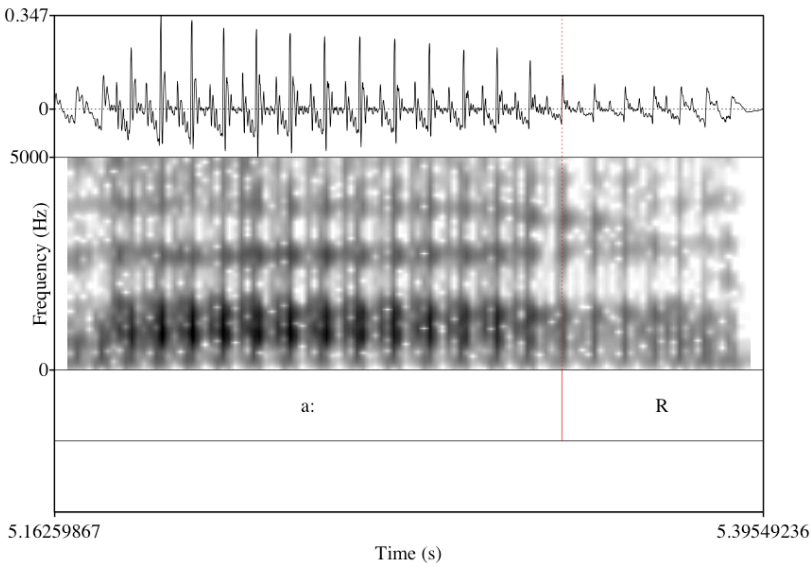

Abbildung 3.2.20.: [aːʁ] in der Folge /faːrt/ – US10020004

Realisierung ist die Friktion aber oft schwach oder nicht vorhanden. Im häufigen letzteren Fall ist eine Einordnung als Approximant [ʊ] phonetisch angemessener. Die Abbildungen 3.2.22 und 3.2.23 zeigen beide Realisierungen.

Die Phonemfolge /ən/, die als Flexionsmorphem in Infinitiven und Substantivpluralendungen vorkommt, wird in der Regel silbisch als [n̩] realisiert. Bei besonders

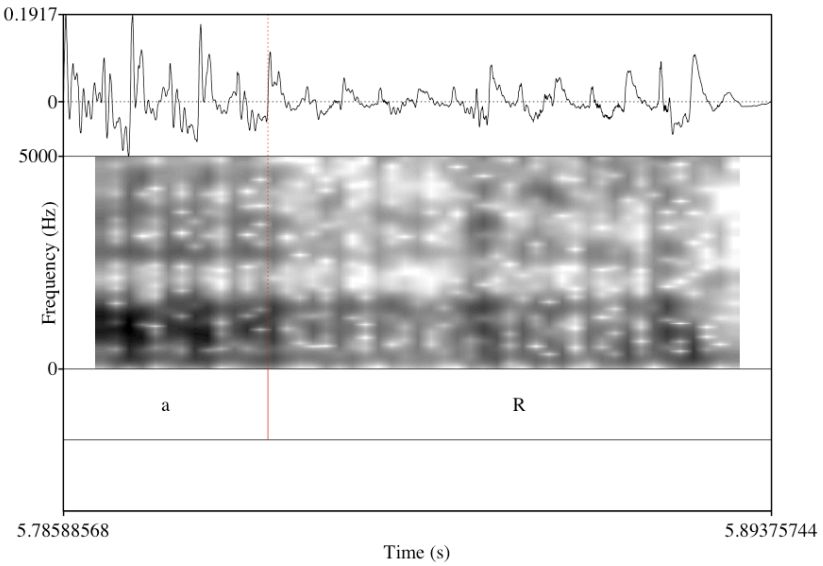

Abbildung 3.2.21.: [aʁ] in der Folge /mar.t/ – US10020004

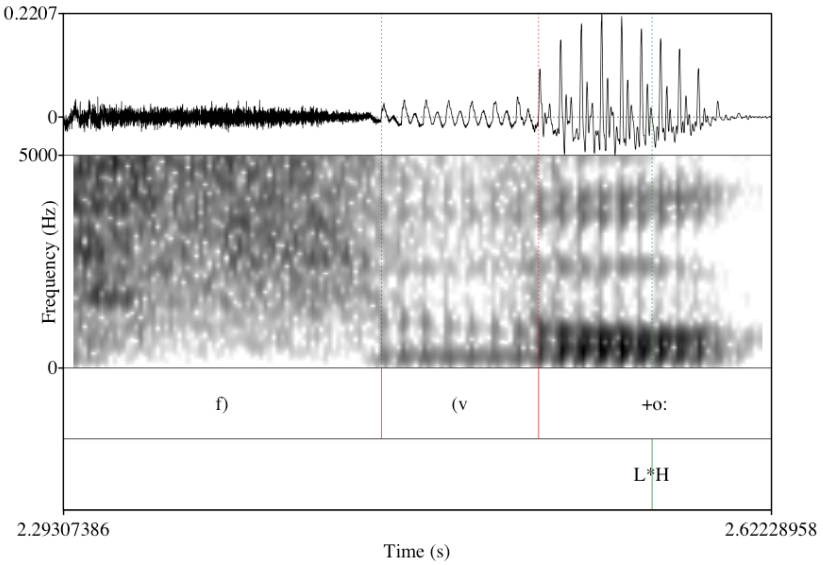

Abbildung 3.2.22.: [voː] in der Folge /f voːk/ – US10020005

deutlicher Sprechweise kann das Schwa auch artikuliert werden oder sogar hyperkorrekt durch den Vokal [ɛ] ersetzt werden. Teilweise wird das Schwa auch nur noch als Rudiment realisiert. Abbildung 3.2.24 zeigt zunächst den optimal segmentierbaren Fall eines Übergangs [ə.n] an der Silbengrenze. In den Abbildungen 3.2.25 bis 3.2.28 werden dagegen die Vorkommen illustriert, die zu einer Einheit zusammengefasst werden.

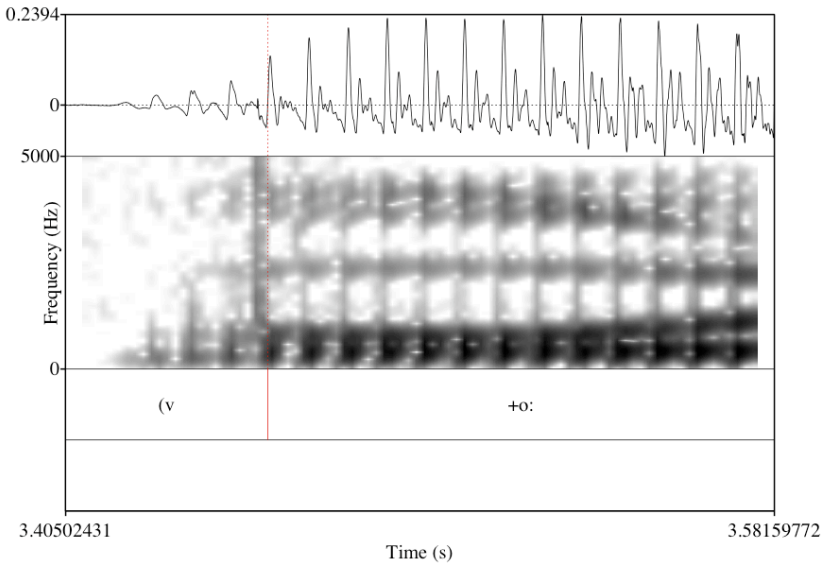

Abbildung 3.2.23.: [ʊoː] in der Folge /k voːl/ – US10020249

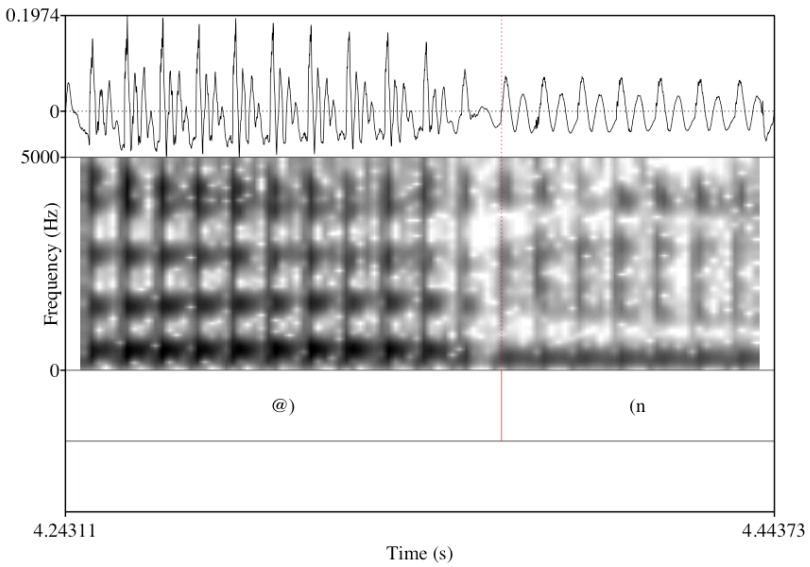

Abbildung 3.2.24.: [ə.n] in der Folge /sə nɔʏ/ – US10020017

3.2.2. Entwicklung und Spezifikation

Eine erste Version der Phoxsy-Einheitendefinition wurde 2001 unter dem Namen BLF (*boss label format*) präsentiert (Breuer et al.). Sie fand zunächst Anwendung bei der Annotation des Korpus für die Telefonauskunftssynthese (s. Kapitel 6). Später wurde zur Abgrenzung der Einheitendefinition vom Dateiformat *boss label file* (eben-

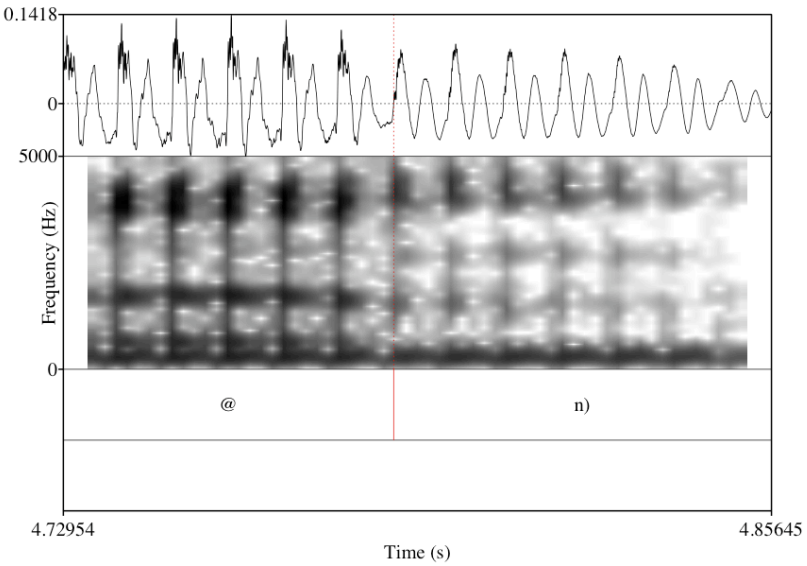

Abbildung 3.2.25.: [ɛn] (nicht als solches annotiert) in der Folge /nən p/ – US10020030

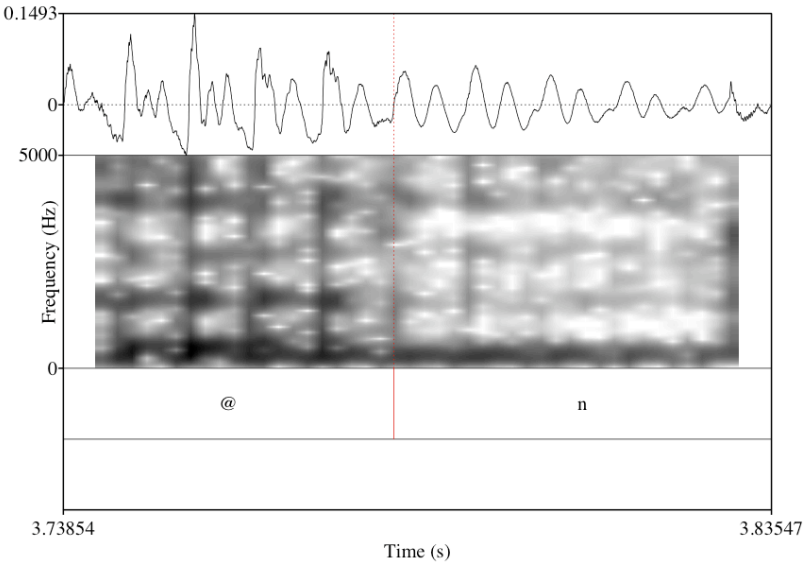

Abbildung 3.2.26.: [ən] (gut separierbar) in der Folge /tseːn.h/[6]– US10020014

falls BLF, vgl. Abschnitt 4.1.1) die Umbenennung in Phoxsy vorgenommen (Breuer und Abresch, 2004). Die aktuelle Fassung enthält die im vorangegangenen Abschnitt aufgezählten Fälle als Multiphone. Zusätzlich wurden die in boss-sampa enthaltenen fremdsprachlichen (oder xenophonen) Approximanten mit allen möglichen Folgevokalen in die Phoxsy-Spezifikation mit aufgenommen. Abbildung 3.2.29 gibt einen Überblick über alle möglichen Multiphone aus Anlaut-Vokal-Kombinationen. Für

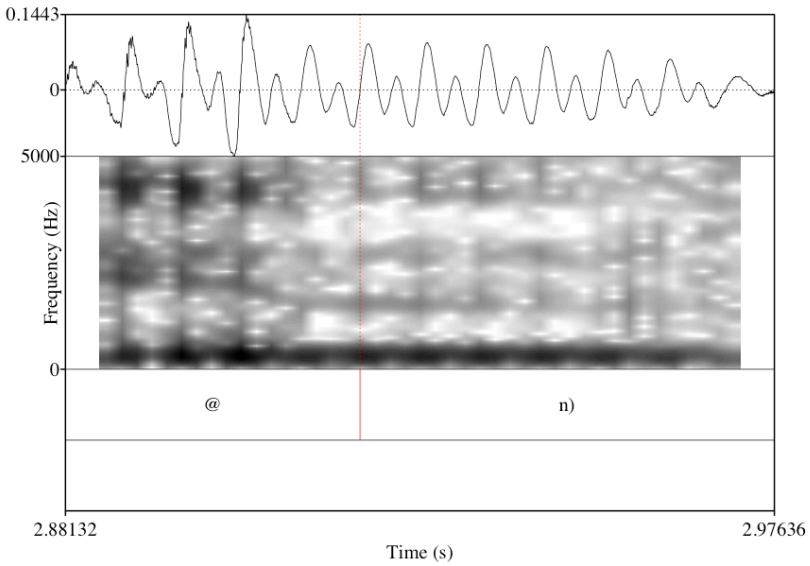

Abbildung 3.2.27.: [ən], ([ə] nur rudimentär vorhanden) in der Folge /gən f/ – US10020020

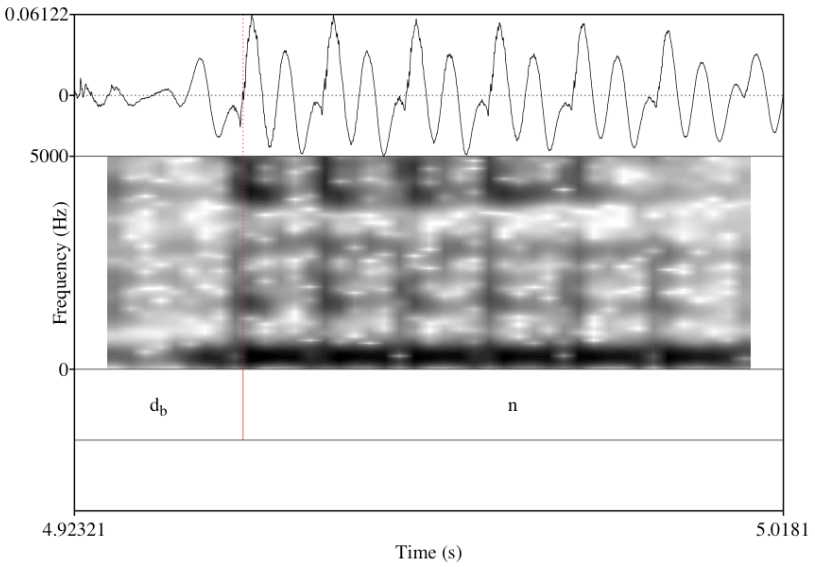

Abbildung 3.2.28.: [n̩] in der Folge /dən.d/ – US10020001

den praktischen Einsatz wurde diese Zahl nach Plausibilitätskriterien beschränkt (s. Abschnitt 6.2.2).

Die Zusammenfassung von /ən/-Realisierungen zu Multiphonen wurde auf alle Kombinationen von Schwa mit silbisch realisierbaren Nasalen und Liquiden jeweils mit und ohne die obengenannten Anlaute verallgemeinert. Die resultierenden Einheiten

IPA	phoxsy
j/ʲ/ç/i̯	j
l	l
ʁ/ʀ/ʱ/ʵ(χ)	r
h/ɦ	h
ʊ/ʋ	v
ɹ	R
w/o̯/ɥ	w
ʔ	?

(a) Konsonanten

×

IPA	phoxsy	IPA	phoxsy
iː	i:	iːɐ	i:6
ɪ	I	ɪɐ/ʁ/ʀ/ʱ/ʵ	I6
yː	y:	yːɐ	y:6
ʏ	Y	ʏɐ/ʁ/ʀ/ʱ/ʵ	Y6
eː	e:	eːɐ	e:6
ɛ	E	ɛɐ/ʁ/ʀ/ʱ/ʵ	E6
ɛː	E:	ɛːɐ	E:6
ø	2:	øːɐ	2:6
œ	9	œ(ɐ/ʁ/ʀ/ʱ/ʵ)	96
ə	@	aɪ	aI
ɐ	6	aʊ	aU
a	a	aɐ/ʁ/ʀ/ʱ/ʵ	a6
aː	a:	aːɐ	a:6
ɔ	O	ɔɐ/ʁ/ʀ/ʱ/ʵ	O6
ɔː	O:	ɔːɐ	O:6
oː	o:	oːɐ	o:6
ʊ	U	ʊ͡ɐ/ʁ/ʀ/ʱ/ʵ	U6
uː	u:	uːɐ	u:6
ɛ̃	E~	eɪ/ɛɪ	eI
ɔ̃	O~	oʊ/ɔʊ/əʊ	oU
ɑ̃	A~	ʊɪ	UI

(b) Monophthonge und Diphthonge

Abbildung 3.2.29.: Multiphoneinheiten aus Konsonant-Vokalkombinationen in phoxsy.

sind in Abbildung 3.2.30 dargestellt. Da jede Einheit verschiedene Realisierungen einer bestimmten Tiefenstruktur repräsentiert, dürfen z. B. Assimilationen von [n̩] zu [m̩] nicht mit dem Label `@m` versehen werden, sondern müssen als `@n` etikettiert bleiben.

Auch von diesen Zusammensetzungen sind bei weitem nicht alle phonotaktisch plausibel oder sehr wahrscheinlich. Für die initiale Anwendung der Eigennamensynthese (s. Kapitel 6) wurden jedoch zur Sicherheit alle Kombinationen in die Spezifikation mit aufgenommen.

3.2.3. Kontextklassen

Durch die Einführung der Multiphone steigt die Anzahl der Einheitentypen im Korpus plus dem leeren Kontext (`$p`) von 50 auf 411 an. Wenn die Obergrenze der

IPA	phoxsy
j/ʲ/ç/i͡	j
l	l
ʁ/ʀ/ʁ̞/ʁ̝(χ)	r
h/ɦ	h
ʊ/v	v
r	R
w/o͡/ɥ	w
ʔ	?

(a) Konsonanten

×

IPA	phoxsy
əl/l̩	@l
ən/n̩	@n
əm/m̩	@m

(b) Schwa-Endsilben

Abbildung 3.2.30.: Multiphone aus Schwa plus silbisch realisierbaren Nasalen und Liquiden — mit und ohne Anfangskonsonanten.

Anzahl möglicher Kombinationen mit linken und rechten Kontexten (Triphonkontexten) als n^3 angenommen wird[7], besteht kaum Hoffnung, ein Korpus aufnehmen und annotieren zu können, das selbst ohne Berücksichtigung prosodischer Varianten alle Einheiten in allen Kontexten enthält (insgesamt 69.426.531 statt 125.000 Triphone). Dies ist aber auch nicht unbedingt nötig, da die linken und rechten Kontexte vereinfacht auch als äquivalent mit den in ihnen enthaltenen unmittelbaren Vorgänger- bzw. Nachfolgerlauten des mittleren Phons angesehen werden können. Ein gesuchter Triphonkontext [ja: ja: ja:] würde dadurch in der Einheitenauswahl auf die Folge [a: ja: j] abgebildet. Dies ist zwar im Hinblick auf die Koartikulation nicht optimal, die Anzahl benötigter Kombinationen sinkt damit jedoch auf reellere $b^2 \cdot n$, (1.027.500) wenn b die Anzahl der boss-sampa-Typen ohne Multiphone ist. Durch eine zweite Stufe der Zusammenfassung, die nur noch den Artikulationsort des Kontextes beschreibt, ist sogar eine Reduktion auf 59.184 Triphonkontexte möglich. Alle Abbildungsvorschriften von Einheiten auf Kontextklassen für das deutsche Phoxsy finden sich in Tabelle A.2.1 im Anhang. Die Implementierung in BOSS ist in Abschnitt 5.3.1 beschrieben.

3.3. Evaluation durch Präferenztests

Um abzuschätzen, ob sich der Einsatz der Phoxsy-Einheiten auf die Qualität der Synthese auswirkt, wurde ein Experiment durchgeführt, in dem den Versuchspersonen jeweils Phoxsy- und boss-sampa-Versionen einer Äußerung präsentiert wurden (Breuer und Abresch, 2004). Die Teilnehmer mussten sich dann für die von ihnen

[7] — wenngleich darin auch praktisch unmögliche Folgen enthalten sind —

nach einem globalen Qualitätskriterium präferierte Variante entscheiden. 49 Stimuluspaare wurden jeweils zweimal – in unterschiedlicher Reihenfolge – abgefragt. Die Abfolge der Stimuluspaare wurde für jede Versuchsperson zufällig ausgewählt. Es nahmen neun Personen mit eigener Forschungserfahrung im Bereich Phonetik und Sprachsynthese an dem Experiment teil. Für die Durchführung des Präferenztests wurde eine in Java entwickelte Testsoftware eingesetzt.

3.3.1. Auswahl der Stimulusvorlagen

Für die Evaluation wurde das Verbmobil-Korpus verwendet, da dieses, im Unterschied zum Eigennamen-Korpus für die Telefonauskunft (s. Kapitel 6) auch über eine Segmentierung in Einzelphone verfügte. Einige der dargestellten Multiphone wurden bereits für die Verbmobil-Synthese verwendet. Im Einzelnen waren dies `@n`, die Vokal-/r/-Kombinationen und die Glottalverschlussfolgen, so dass diese von dem Vergleich nicht erfasst wurden.

Der Einfachheit halber sollten Sätze als Stimuli verwendet werden, die bereits im Korpus vorhanden waren, um sie dann unter Auslassung der im jeweiligen Satz vorkommenden Syntheseeinheiten zu resynthetisieren (*leave-one-out*). Zur Gewinnung der Teststimuli wurden zunächst alle Sätze aus dem Synthesekorpus ausgewählt, die Phoxsy-Einheiten enthielten. Um zu gewährleisten, dass eine Resynthese nach Entfernen des Satzes aus dem Korpus noch möglich ist, wurde überprüft, ob die enthaltenen Phoxsy-Typen jeweils noch in mindestens einem anderen Satz enthalten waren. Jedem Satz, auf den diese Bedingung zutraf, wurde dann durch Anwendung von Formel 3.3.1 eine Punktzahl zugewiesen, die sich aus der Multiplikation der Anzahl der Phoxsy-Einheiten p mit dem Verhältnis zwischen Phoxsy- und Einzelphon-Einheiten s, addiert zur Anzahl der unterschiedlichen Phoxsy-Typen im Satz d, ergibt.

$$\text{score} = \frac{p^2}{s} + d \tag{3.3.1}$$

Beginnend mit dem Satz, der die nach diesem Verfahren höchste Bewertung erzielte, wurde der jeweils nächstbeste Satz automatisch selektiert, bis eine Menge von 50 Testsätzen ausgewählt war. Dabei wurden alle Sätze ausgelassen, die mehr als drei Phoxsy-Einheiten enthielten, um die Länge der Sätze zu begrenzen. Zudem wurden nur solche Sätze gewählt, die mindestens einen Einheitentyp hinzufügten, der in der bisherigen Auswahl noch nicht enthalten war.

Dieses Verfahren entspricht einem Greedy-Algorithmus zur Lösung eines Mengenüberdeckungsproblems, wie er häufig auch zur Auswahl von Korpuslesetexten für die Synthese zum Einsatz kommt (s. Abschnitt 2.5.3), und der bei einer möglichst

guten Abdeckung von Varianten eine möglichst geringe Textmenge auswählen soll.

3.3.2. Generierung der Stimuli

Die ausgewählten Testsätze wurden in einer Phoxsy- und einer boss-sampa-Version unter Auslassung der Einheiten des betreffenden Korpussatzes synthetisiert. Um eine solche Resynthese zu ermöglichen, musste der BOSS-Quelltext zunächst um eine entsprechende Funktion erweitert werden.

Die Resynthese-Funktion, die für das Pilotexperiment geschrieben wurde, liest eine Textdatei ein, die pro Zeile einen Dateinamen eines Korpussatzes enthält. Zu jedem dieser Sätze werden für alle Einheitentypen (`WORD`, `SYLLABLE`, `PHONE`, `HALFPHONE`) die in der Datenbank gespeicherten Annotationsdaten abgerufen. Diese können dann verwendet werden, um die Ausgaben der symbolischen Vorverarbeitungsstufen (hier Transkription und Dauerprädiktion) durch die wirklichkeitsgetreuen Werte aus den tatsächlichen Sprachdaten zu ersetzen. Für das beschriebene Experiment war dies notwendig, um identische Vorgaben für die Einheitenauswahl von Phoxsy- und boss-sampa-Einheiten zu erhalten, da ja zunächst nur die Eignung der verschiedenen Einheiten per se ermittelt werden sollte.

Attribut	Funktion	PS	US
`TKey`	Transkription der Einheit	•	–
`CLeft`	linker Lautkontext der Einheit	•	–
`CRight`	rechter Lautkontext der Einheit	•	–
`CCLeft`	1. Kontextklasse des linken Lautkontextes	•	–
`CCRight`	1. Kontextklasse des rechten Lautkontextes	•	–
`CCLeft2`	2. Kontextklasse des linken Lautkontextes	•	–
`CCRight2`	2. Kontextklasse des rechten Lautkontextes	•	–
`PMode`	Phrasengrenztyp	–	20.0 *(ucw_phrase)*
`PInt`	Phrasengrenztonstärke	•	–
`Dur`	Dauer der Einheit	–	1.0 *(ucw_dur)*
`Stress`	lexikalische Akzentstufe der Einheit	•	10.0 *(ucw_stress)*
`ResynthNum`	eindeutige Nummer der Einheit in der Datenbank	•	–

Tabelle 3.3.1.: Zur Resynthese verwendete Attribute und ihre Berücksichtigung in den verschiedenen Stufen der Einheitenauswahl. Die Zahlen stellen die Gewichtungsfaktoren der Einheitenkosten (unit cost weights, *ucw*) dar. PS=Pre-Selection, US=Unit-Selection

Tabelle 3.3.1 beschreibt, in welcher Form welche Attribute zur Vorauswahl (s. Abschnitt 6.3.3) und kostenbasierten Auswahl (s. Abschnitt 5.3.3) von Einheiten eingesetzt wurden. Das in der Standardversion von BOSS nicht vorhandene Attribut `ResynthNum` bezeichnet die eindeutige Nummer einer Originaleinheit und entspricht den Indizes in der `Num`-Spalte der Datenbanktabellen. Diese Nummer wird im für das Experiment angepassten Unit-Selection-Modul verwendet, um sicherzustellen, dass das entsprechende Segment nicht für die Auswahl verwendet wird. Um einen direkten Vergleich zwischen phonbasierter und Multi-Phon-basierter Synthese zu ermöglichen, wurde die Vorauswahl so konfiguriert, dass Wörter, Silben und Halbphone nicht als Syntheseeinheiten gewählt werden konnten. Die Transitionskosten der Kandidateneinheiten mit den benachbarten Kandidaten wurden wie in Abschnitt 5.3.3 beschrieben berechnet. Für die Gewichtungsparameter *tcw_dur* und *tcw_f0* wurde jeweils der Wert 1.0 eingestellt.

3.3.3. Analyse der Stimuli

Einer der synthetisierten Sätze besaß eine fehlerhafte Transkription, so dass das entsprechende Stimuluspaar aus den Testdaten entfernt wurde. Damit standen für den Test 49 Stimuluspaare zur Verfügung. Da eine zufällige Synthese beider Versionen eines Stimuluspaares aus den gleichen Signalabschnitten im Originalkorpus nicht verhindert werden konnte ohne das Ergebnis des Einheitenauswahlverfahrens zu verfälschen, wurden keine Maßnahmen getroffen, um eine solche Übereinstimmung zu vermeiden. Eine Analyse der 98 synthetischen Stimuli zeigte jedoch, dass nur 28 der 146 enthaltenen Phoxsy-Einheiten ein exaktes boss-sampa-Gegenstück besaßen. Tabelle 3.3.2 zeigt die verwendeten Phoxsy-Einheiten.

Stimuluspaar	Einheiten	Stimuluspaar	Einheiten
1	**hE6 ja:** rOY	**25**	**hE6 l@n** lu:
2	ja: **vo:**	**26**	**jE** ra **vo:**
3	*ja:* lo: r@n	**27**	**haI raU ry:**
-	entfernt	**28**	**raI ro: vE6**
4	**ha ja:** lo:	**29**	**lE rI vi:6**
5	**ha lo: raI**	**30**	**ha: la vo:**
6	**hE6 ja: lO**	**31**	**ha: hi:6 la:**
7	**ja:** rU **va**	*32*	*ja:* laI *v9*
8	**ju: li:** rI	**33**	**ha: hi:6 ly:**
9	**hE6 ja:** ra:	**34**	**ju:6 raU vi:**

10	*r@* rE *vE:*	**35**	**hE jE va:**
11	l@ la: **ra:**	**36**	**h2:6 ha: ja:**
12	*ha: vE vi:6*	37	**lu: raU** rY
13	**ry: va va:6**	**38**	**lOY ra ro:**
14	**ju: vaI vO**	**39**	**hi:6 r@ re:**
15	*lO raU vi:*	*40*	*hI ja: vY6*
16	*hi: ho: vi:*	41	ho: laU **rO**
17	**ja:** le: **ro:**	42	l@n **r@n** ru:
18	**hI ri: vi:6**	*43*	*ha: j@n* rI
19	**ja:** laI ry:	*44*	*ja: raU ve:*
20	*l6 raU vi:*	45	**ja rE** vi:6
21	**raU vi: vI**	*46*	*ra: rI vU6*
22	laI *l@n vi:6*	*47*	*ja:* lI *raI*
23	**haU hI vI6**	48	**ja** lU **vi:6**
24	*ja: jE ve:6*	49	**ha ho:** vOY

Tabelle 3.3.2.: Stimuluspaare und zugehörige phoxsy-Einheiten:*Sätze und Einheiten, die inkonsistent beurteilt wurden*; phoxsy-Einheiten, die aus den gleichen Korpuseinheiten zusammengesetzt wurden wie die boss-sampa-Gegenstücke (auch bei inkonsistenter Beurteilung schwarz gesetzt); **phoxsy-Einheiten, die aus verschiedenen Einheiten zusammengesetzt wurden.** Nach Breuer und Abresch (2004).

3.3.4. Ergebnisse

Eine erste Analyse der Resultate aller Versuchspersonen für alle Stimuluspaare zeigte eine signifikante Präferenz von 54 Prozent für die mit Phoxsy-Einheiten synthetisierten Stimuli. Alle Probanden berichteten jedoch von Schwierigkeiten bei der Festlegung auf eine präferierte Variante aufgrund der hohen akustischen Ähnlichkeiten. Diese Unsicherheit spiegelte sich in einigen Fällen auch in einer starken Inkonsistenz der Entscheidungen für die gleichen Stimuluspaare in unterschiedlichen Reihenfolgen (A-B/B-A) wider. Auch wiesen drei der Probanden eine durchgängig schwache Konsistenz ihrer Entscheidungen auf (< 60 Prozent). Aus diesen Gründen wurden die Ergebnisse in die in Tabelle 3.3.3 dargestellten Ergebnismengen aufgeteilt und

individuell analysiert.

Menge	N Phoxsy / sampa	χ^2	p
ALL SUB - ALL STIM	475 / 407	5,243	0,022
CON SUB - ALL STIM	332 / 256	9,823	0,002
ALL SUB - CON STIM	348 / 281	7,340	0,07
CON SUB - CON STIM	**243 / 177**	**10,371**	**0,001**
INC SUB - ALL STIM	143 / 151	0,218	0,641
INC SUB - CON STIM	106 / 104	0,19	0,890

Tabelle 3.3.3.: Signifikanzwerte für verschiedene Untermengen der Ergebnisse (CON = konsistent, INC = inkonsistent, SUB = Probanden, STIM = Stimuli). Nach Breuer und Abresch (2004).

Dabei zeigte sich, dass die Menge der Entscheidungen der konsistent antwortenden Probanden zu den allgemein mit größerer Konsistenz beurteilten Stimuluspaaren mit 58 Prozent die stärkste Präferenz von Phoxsy-Einheiten aufweisen. Dies basiert auf 420 Beurteilungen von 49 verschiedenen Phoxsy-Einheiten in 34 Sätzen durch sechs Versuchspersonen. Die statistischen Analysen in Tabelle 3.3.3 unterstützen die Annahme, dass diese Menge den Antworten

- nur für die unterscheidbaren Stimuli und
- derjenigen Versuchspersonen entspricht, die überhaupt in der Lage waren, eine Unterscheidung zu treffen.

3.4. Diskussion

Auch wenn die Ergebnisse deutlicher sein könnten, sie zeigen, dass es sich auszahlen kann, die beschriebene Multiphondefinition in der Synthese einzusetzen und dass eine genauere Analyse der Umstände lohnend wäre, unter denen die Phoxsy-Einheiten einen qualitativen Vorsprung gegenüber der phonbasierten Segmentierung besitzen. Insbesondere der Einfluss auf die prosodische Beurteilung von Sätzen wäre hier von Interesse, aber auch die globale Verständlichkeit, sowie die Verstehbarkeit einzelner Einheitentypen sollten genauer betrachtet werden.

Bei der Bewertung der Resultate muss beachtet werden, dass eine nicht unwesentliche Untermenge von Phoxsy schon in der Verbmobil-Annotation enthalten war, so dass ein Vergleich mit einer vollständigen Einzelphonannotation die Unterschiede stärker herausstellen könnte.

Es ist möglich, dass die längeren Ketten aufgrund ihrer inhärenten prosodischen Eigenschaften bevorzugt wurden. Der Grundfrequenzverlauf der Ursprungsäußerung war kein Kriterium für die Resynthese, so dass sich die Intonation rein aus der Vorgabe der lexikalischen Akzentuierung und der Phrasierung ergeben musste. Obwohl ein Kontinuitätskriterium für den F0-Verlauf in den Transitionskosten verwendet wurde und obwohl für die Multiphone inhärent weniger prosodische Varianten zur Verfügung stehen, könnten hier die zusammenhängenden Verläufe im Vorteil sein. Sollte dies ein Kriterium für die Bevorzugung sein, wäre zumindest der Einsatz in Systemen angezeigt, die keine umfassende prosodische Modellierung vornehmen und zu denen auch BOSS momentan noch gehört (vgl. 5.5). Phoxsy wäre wegen der zeitlichen Einsparungen bei der Annotation und Auswahl allerdings auch dann schon generell zu empfehlen, wenn die Qualität nicht besser wäre als die der Einzelphonsynthese.

Natürlich wird hier nicht der Anspruch erhoben, dass durch die präsentierten Multiphon-Typen alle Phänomene abgedeckt werden, die zur schlechten Segmentierbarkeit von Einheiten führen können oder in denen Auslassungen auftreten. Dies wird aber zumindest an den wichtigsten Stellen geleistet, an denen auch bei sorgfältiger Sprache starke Variation bis hin zur Elision von Segmenten auftreten kann. Für Syntheseszenarien, die auf schnellere und/oder spontansprachliche Ausgaben angewiesen sind, müssen ggf. weitere Zusammenfassungen vorgenommen oder auf eine zusätzliche Modellierung der Aussprache zurückgegriffen werden. Dies wäre allerdings nur unter der bereits genannten Prämisse ratsam, dass die verwendeten Korpora phonetisch sehr eng transkribiert sind. Die Eignung von speziellen Multiphon-Einheiten für die schnelle Ausgabe durch Vorleseautomaten für Blinde wird in einem derzeit am IfK laufenden Dissertationsvorhaben von Moers untersucht (Moers et al., 2007).

Der Phoxsy-Ansatz stellt keinen Widerspruch zu den zuvor dargestellten Einheitendefinitionen und Auswahlmechanismen dar. Obwohl segmentell definiert, stellen sie nur eine Untergrenze der sinnvollen Aufteilbarkeit dar. Sie lassen sich jederzeit zu größeren non-uniform units zusammensetzen, aber durch geschicktes Setzen der Schnittpunkte auch in Halbphone und Diphone aufteilen. Dadurch sind sie mit allen vorgestellten wellenformbasierten Auswahlverfahren, inklusive der Auswahl ganzer Phrasenabschnitte, wie beim PSM, kompatibel. Die Aufteilung in Diphone wird sogar empfohlen, weil sie das Problem der großen Anzahl an Triphonkontexten entschärft, und ist daher das nächste Ziel der Entwicklung. Hierbei muss nur darauf geachtet werden, dass die Grenzen der Diphone erst im stetigeren hinteren Teil der Multiphone gesetzt werden, um die Vorteile der Zusammenfassung nicht zu verspielen.

Wenn auch die hier vorgestellte Phoxsy-Definition sprachspezifisch für das Deutsche ist, gilt dies nicht für die Prinzipien, die hinter Phoxsy stecken und noch weniger

für den hier versuchten Anstoß, die Phonetik zumindest an den Schnittstellen zwischen Symbol und Signal wieder mit in den Vordergrund der Syntheseforschung zu stellen. Im Zuge der Anpassung von BOSS an das Polnische wurden bereits erste Multiphon-Einheiten für diese Sprache konzipiert, so dass die Arbeiten zu Phoxsy auch als Beitrag zur multilingualen Verwendung des BOSS-Systems betrachtet werden können. Im folgenden Kapitel werden zunächst die Grundlagen dieses Systems erläutert, um dann in Kapitel 5 auf die softwaretechnischen Entscheidungen zum Ausbau von BOSS zu einer multilingualen und multifunktionalen Synthesearchitektur eingehen zu können.

4. Bonn Open Synthesis System

„Entzwei und gebiete! Tüchtig Wort. – Verein und leite! Besserer Hort."
J. W. v. Goethe

Das *Bonn Open Synthesis System*, kurz BOSS, ist eine von Karlheinz Stöber entwickelte Open-Source-Architektur für die konkatenative Sprachsynthese, insbesondere nach dem Unit-Selection-Verfahren. War die erste Version noch ein direktes Derivat der Verbmobil-Synthese (Stöber et al., 2000; Abschnitt 2.3), inklusive der Blackboard-Architektur zur Modulkommunikation, handelte es sich beim hier beschriebenen BOSS II (Klabbers et al., 2001; Klabbers und Stöber, 2001; Stöber, 2002) um eine komplette Neuimplementierung mit moderneren Datenstrukturen und größerer Flexibilität. Dieses Kapitel beschreibt den grundlegenden Aufbau der Architektur und die Konzepte für die Einheitenauswahl und Korpusannotation, die den Ausgangspunkt für die im nächsten Kapitel beschriebenen eigenen Erweiterungen und Umstrukturierungen hin zu Version 3 und darüber hinaus bildeten.

4.1. Korpuserstellung und -repräsentation

Außer den unmittelbar für die Synthese genutzten Programmteilen besteht die BOSS-Architektur aus einer Reihe von Werkzeugen zur Korpusannotation. Für die phonetisch-akustische Beschreibung von Sprachkorpora existieren in BOSS drei Repräsentationsebenen, die unterschiedlichen Anforderungen gerecht werden. Im Folgenden sollen diese Formate und die Tools, die sie erzeugen, näher beschrieben werden.

4.1.1. BOSS Label File

BOSS Label Files (BLF) sind einfache, zeilenbasierte Darstellungen einer Lautfolge in einem Sprachsignal. Sie dienen zur Betrachtung und manuellen Annotation der Korpussegmentierung. Der Audio-Editor *wavesurfer* (Sjölander und Beskow, 2000) unterstützt das Lesen und Schreiben von BOSS-Labeldateien. Abbildung 4.1.1 zeigt

links ein BLF für die Ein-Wort-Äußerung ['peː.tɐ]. Das Format ist dreispaltig. In der ersten Spalte befinden sich die Sample-Werte, die den Beginn einer Einheit in der Signaldatei markieren. Die zweite Spalte enthält die symbolische Beschreibung des Lautes in boss-sampa (Abschnitt 3.2) inklusive lexikalischer Akzente und Wort- und Silbengrenzmarkern. Nur an Phrasengrenzen wird die dritte Spalte benötigt. Der erste Laut eines Wortes an einer solchen Grenze erhält in dieser Spalte einen Eintrag in dem in Tabelle 4.1.1 dargestellten Format. Die beiden Parameter `PInt` und `PMode` werden dabei durch ein Komma getrennt. Die Tabelle zeigt den Wertebereich der Attribute und ihre Verwendung in der deutschen Synthese.

PMode (`?` oder `.`)	PInt (−`5`...`5`)	Bedeutung
`.`	`5`	Aussageintonation am Satzende
`?`	`5`	Frageintonation am Satzende
`?`	`2`	Kontinuität an intermediärer Phrasengrenze

Tabelle 4.1.1.: Wertebereiche der Attribute `PMode` und `PInt` zur Beschreibung von Phrasengrenzen und die für die deutschen Datenbanken genutzten Kombinationen.

`PMode` ist dabei eine abstrakte Beschreibung der Richtung der F0-Bewegung, während `PInt` die Stärke der Grenze repräsentiert.

4.1.2. BOSS-Inventar-XML

Die weitergehende, automatische Annotation der Sprachsignaldateien mit zusätzlichen Parametern wird nicht mehr mit Hilfe von BLF vorgenommen. Hierfür sind XML-Dateien in einem spezialisierten BOSS-Vokabular vorgesehen, die durch das Programm `blf2xml` aus den Labeldateien erzeugt und dann schrittweise durch weitere Tools mit Informationen versehen werden. Ein Vorteil dieser Darstellung ist, dass für das Parsing keine proprietären Eigenimplementierungen notwendig sind, sondern auf standardisierte XML-Parser, hier *xerces-c* (The Apache Software Foundation, 2007), zurückgegriffen werden kann.

Abbildung 4.1.1 zeigt rechts ein dem gezeigten BLF entsprechendes *BOSS-Inventar-XML*. Das XML stellt die Informationen aus dem Labelfile in einer hierarchischen linguistischen Struktur dar. Auf der obersten Ebene befindet sich der Satz, der auch dem maximalen Skopus einer BLF-Datei entspricht. Er wird durch das Element `<SENTENCE>` repräsentiert. Darunter befinden sich die Ebenen `<WORD>`, `<SYLLABLE>` und `<PHONE>`[1]. Auf jeder Ebene werden so viele Elemente eines Typs erzeugt, wie es

[1] Ursprünglich: `<PHONEME>`.

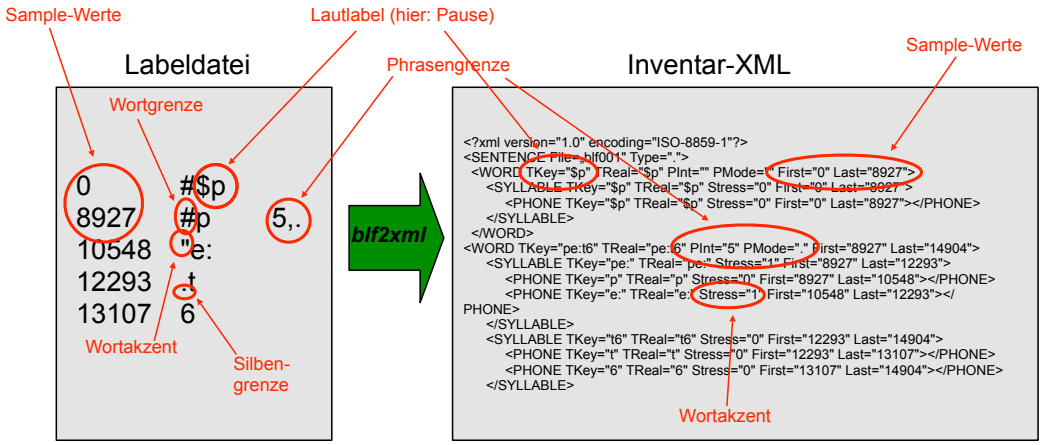

Abbildung 4.1.1.: Beziehung zwischen der Repräsentation einer Signalannotation durch BLF und der XML-Kodierung.

entsprechende Einheiten in der BLF-Quelldatei gibt. Das bedeutet, es wird für jedes Wort in der Eingabedatei ein Element `<WORD>` erzeugt, das wiederum die jeweils zugehörigen `<SYLLABLE>`-Elemente enthält, welche ihrerseits die ihnen untergeordneten `<PHONE>`-Elemente beinhalten. Die Eigenschaften eines jeden Elements werden durch seine Attribute beschrieben, während die Elemente selber keine Informationen enthalten. Mit Ausnahme des Satzes werden in diesem ersten Vorverarbeitungsschritt für alle Elemente die gemeinsamen Attribute `TKey` (kanonische Transkription) und `TReal` (realisierte Aussprache bei manueller Annotation), sowie `First` und `Last` (Start und Endzeitpunkt der Einheit) vergeben. Auf der Ebene des Wortes kommen noch die Phrasierungsattribute `PInt` und `PMode` hinzu; Silben- und Lauteinheiten erhalten zusätzlich das Attribut `Stress`, um die An- oder Abwesenheit lexikalischer Akzentuierung anzuzeigen. `<SENTENCE>` enthält lediglich das Attribut `File`, das den Namen der Quelldatei angibt, sowie `Type` zur Unterscheidung von Satzmodi[2].

Abbildung 4.1.2 zeigt den weiteren Ablauf der Korpuserstellung in BOSS II (Schritte 2–4). Zunächst wird mit dem Programm `addcontext` jedem Wort-, Silben- und Phon-Element die Information über seinen linken und rechten segmentalen Kontext hinzugefügt, indem der Inhalt des `TKey`-Attributes benachbarter `<PHONE>`-Elemente als Attribut `CLeft` bzw. `CRight` gespeichert werden. Dann wird das Programm `optbounds` verwendet, um die Start- und Endzeitpunkte `First` und `Last` aller Einheiten so anzupassen, dass sie in einem Nulldurchgang der Amplitude mit anschließendem positiven Ausschlag liegen. Mit dem `melbounds`-Tool werden im letzen Annotationsschritt die spektralen Eigenschaften der Einheitengrenzen mit je 11

[2]Dieses Attribut ist durch die `PMode/PInt`-Beschreibung redundant und wird für die deutsche Synthese nicht genutzt.

Mel-Cepstrum-Koeffizienten (Mermelstein, 1976) beschrieben. Dabei werden die Attribute `LM0`, ..., `10` für die linke Grenze und `RM0`, ..., `10` für die rechte eingefügt. Seit der Vorstellung von BOSS II durch Klabbers und Stöber (2001) und Klabbers et al. (2001) wurden Erweiterungen der bestehenden Tools und neue Annotationsschritte entwickelt, die z. T. in den Kapiteln 5 und 6 beschrieben werden. Die Attribute und Strukturen der Inventar-XML-Dateien in der aktuellen Version von BOSS 3 werden im Anhang durch das Schema in Listing C.1 beschrieben.

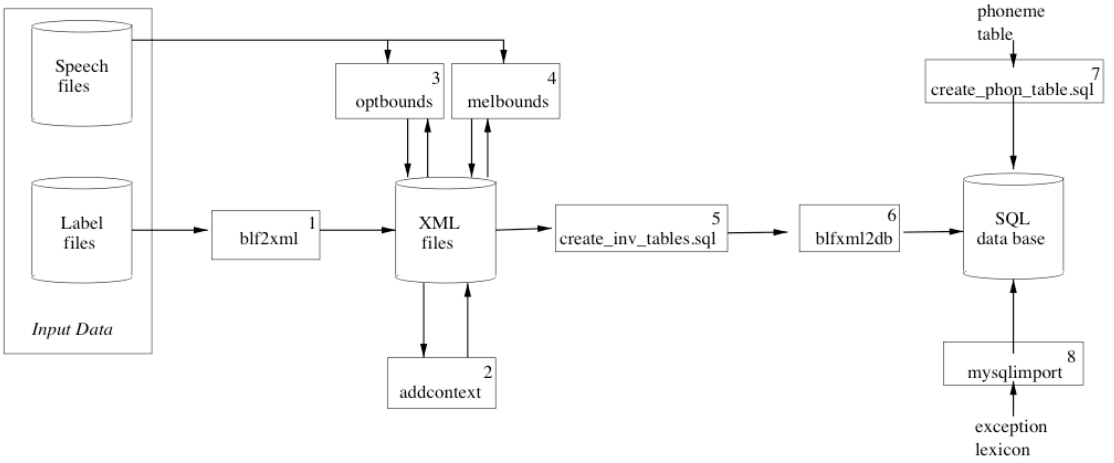

Abbildung 4.1.2.: Ablauf der Korpusaufbereitung in BOSS (Klabbers und Stöber, 2001).

Um das vollständig annotierte Korpus in der Synthese nutzen zu können, muss es zunächst in eine weitere Darstellungsform überführt werden. Zwar bietet sich XML für die oben geschilderten Schritte an, weil es auf standardisierte Formate und Methoden zurückgreift und leicht zu konvertieren ist, der Zugriff zur Laufzeit durch BOSS ist aber u. U. nicht schnell genug möglich. Daher wird die Korpusbeschreibung in einem letzten Schritt in einer relationalen Datenbank gespeichert.

4.1.3. Relationale Datenbankstruktur

Seit Version II verwendet BOSS den MySQL-Datenbankserver und das zugehörige C-API (MySQL AB, 2008), um zur Laufzeit Annotationsdaten abzufragen. Für jede linguistische Elementebene im XML wird eine Tabelle in der Datenbank angelegt (`sentence_data`, `word_data` usw.). Ein Einzelelement (*token*) entspricht einer Zeile in einer solchen Tabelle. Es erhält eine über alle Einheitenebenen einzigartige Indexnummer. Die Informationen der Attribute werden in den Spalten der Tabellen abgelegt. Um die Beziehung zwischen den Ebenen zu erhalten, werden Abbildungstabellen erzeugt, die für jede Einheit eine Beziehung zwischen ihrem Index und den

Indizes der ihr untergeordneten Einheiten herstellen. Abbildung 4.1.3 zeigt, wiederum für das Beispiel ['peː.tɐ], die XML-Darstellung und einen Ausschnitt der daraus generierten Tabellen und Einträge.

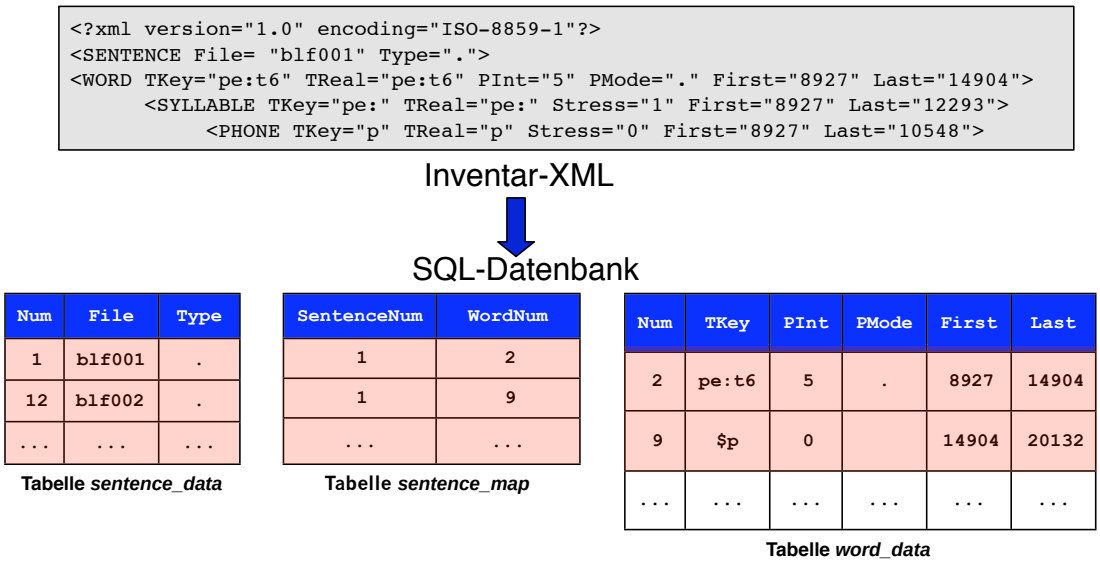

Abbildung 4.1.3.: XML-Kodierung der Annotationsdaten und Darstellung in einer relationalen Datenbank.

Für die Konstruktion der Tabellenstruktur existiert eine SQL-Skriptvorlage[3], die auf das Verbmobil-Korpus zugeschnitten ist. Den Import der Daten aus den XML-Dateien in die Datenbank erledigt das Programm `blfxml2db` (Schritte 5 und 6 in Abbildung 4.1.2).

4.2. Synthesearchitektur

Abbildung 4.2.1 zeigt die Laufzeit-Architektur des deutschen BOSS-II-Systems in der ursprünglichen Version, die das entsprechend aufbereitete Verbmobil-Korpus *lioba* verwendet. Sie besteht aus einer Client- und einer Server-Komponente, der SQL-Datenbank mit den Annotationsdaten und den Sprachsignaldateien. Der Server übernimmt die phonetischen Symbolverarbeitungsschritte (zunächst nur die Transkription), die Einheitenauswahl und die Verkettung der Signale. Die gesamte orthografisch-linguistische Verarbeitung mit Text-Normalisierung und Phrasierung ist hingegen in den Client ausgelagert. Dies soll ermöglichen, wechselweise *text-to-*

[3]Diese wurde seit der hier beschriebenen Version deutlich erweitert. Die SQL-Schemadefinition der Datenbank ist in Listing D.1 im Anhang abgedruckt.

speech (TTS), *content-to-speech*[4] (CTS) oder eine Markup-basierte Zwischenstufe (bspw. mit SSML, s. Abschnitt 4.2.1) ohne weitere Änderungen mit BOSS betreiben zu können. Gemeinsam ist allen Clients das Ausgabeformat *BOSS-Server-XML*, das über die Netzwerk- oder Dateischnittstelle (s. Abschnitt 5.1) an den Server übermittelt wird. BOSS II enthielt einen einfachen generischen TTS-Client für lateinische Schriftsysteme, der in der Dissertation von Stöber (2002) näher beschrieben wird. Ein CTS-Client für die Telefonauskunftsanwendung wird in Abschnitt 6.3.1 vorgestellt.

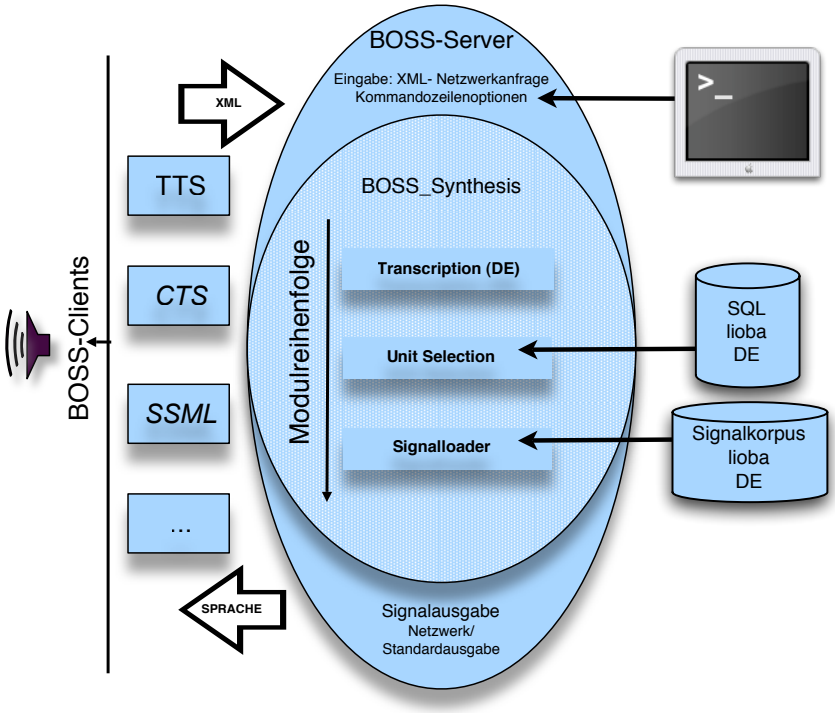

Abbildung 4.2.1.: Architektur des Laufzeit-Systems von BOSS II. Hypothetische Client-Programme sind *kursiv* beschriftet.

4.2.1. XML als Repräsentationssprache

Server-XML-Dokumente werden nicht nur zur Übermittlung von Daten zwischen Client und Server verwendet, sondern dienen auch zur Kommunikation und Datenspeicherung innerhalb des modular aufgebauten Servers. Zur Repräsentation der XML-Struktur im Speicher verwendet BOSS das *Document Object Model* (DOM, Hégaret et al., 2005). Das Vokabular des Server-XML ist in allen wichtigen Eigenschaften identisch mit dem oben beschriebenen Inventar-XML. Da es für die

[4] Auch: *concept-to-speech*; *context-to-speech*.

Repräsentation einer hypothetischen Zieläußerung verwendet wird, fehlen die signalbezogenen akustischen Attribute. Zudem kann es Attribute enthalten, die von einzelnen Syntheseschritten verwendet werden, aber für die Korpusaufbereitung keine Rolle spielen. Das oberste Element einer Server-XML-Datei ist `<TEXT>` und nicht das darunterliegende `<SENTENCE>`, da im Unterschied zum Inventar-XML auch mehrere Sätze in einer Datei beschrieben werden können müssen.

In der Ausgabe des BOSS-Clients existieren noch nicht alle Elementebenen, da die phonetische Struktur der Zieläußerung erst im Server ermittelt wird. Diese unterspezifizierte Form des Server-XML nennt Stöber (2002) *minimales* XML-Dokument. Enthalten sind darin nur die Text-, Satz- und Wortebenen. Da alle Komponenten des BOSS-Servers jedoch nach Möglichkeit vorhandene Elemente und Attribute nicht überschreiben sollen, kann theoretisch auch ein Silben und Phone enthaltendes Dokument an den Server gesendet werden, falls diese Informationen schon im Client ermittelt werden können. Andernfalls werden die für die Synthese benötigten phonetischen Informationen über die Zieläußerung von den einzelnen aufgabenspezifischen Modulen des Servers Schritt für Schritt ergänzt.

```
1 <?xml version='1.0' encoding='ISO-8859-1' ?>
2 <TEXT>
3   <SENTENCE Type=".">
4    <WORD Orth="Peter" PInt="5" PMode="."></WORD>
5   </SENTENCE>
6 </TEXT>
```

Listing 4.1: Beispiel für die Ausgabe des BOSS-Clients: minimales XML-Dokument der Ein-Wort-Äußerung *Peter*.

Da das Server-XML von der abstrakten wortbasierten Darstellung einer Zieläußerung bis hin zur detaillierten phonetischen Beschreibung alle Funktionen wahrnehmen kann, fungiert es im Sinne der Definition von Schröder und Breuer (2004) sowohl als Markup- als auch als Repräsentationssprache. Erstere sind nach dieser Einteilung Sprachen, die für die Eingabe von Text in ein Sprachsynthesesystem geschaffen wurden. Ihr Ziel ist es, auch Laien zu ermöglichen, über den reinen Text hinaus Informationen zur gewünschten Ausgabeform einer Äußerung zu spezifizieren. Dies entspricht bereits einer einfachen Form von CTS. Beispiele für solche Sprachen sind die *speech synthesis markup language*, SSML, (Burnett et al., 2004; Taylor und Isard, 1997) und SABLE (Sproat et al., 1998). Die damit kodierbaren Informationen sind allerdings i. d. R. zu abstrakt, um direkt in der Auswahl- oder Generierungskomponente der Synthese eingesetzt werden zu können. Hier kommen die für die interne Kommunikation der Syntheseanwendung gedachten Repräsentationssprachen ins Spiel. Diese Formate können im Unterschied zu den Markup-Sprachen auch detail-

lierte phonetische und akustische Informationen aufnehmen, die von den Komponenten des Systems selber erzeugt werden und daher in ihrer Darstellung mit den prosodischen und segmentalen Modellen der Einheitenauswahl oder Generierung kompatibel sind. Obwohl systemspezifisch, ermöglichen die XML-Repräsentationssprachen eine einfachere Verbindung zwischen verschiedenen Syntheseanwendungen als andere proprietäre Formate. Schröder und Breuer zeigen dies, indem sie mittels XSL-Transformationen (Clark, 1999) zwischen BOSS-Server-XML und dem vom DFKI-System *MARY* (DFKI, 2008; Schröder und Trouvain, 2003) eingesetzten XML-Format konvertieren und so einerseits die MARY-Dauerprädiktion in BOSS einsetzen und andererseits die Graphem-Phonem-Konvertierung von BOSS (s. Abschnitt 6.3.2.4) für MARY nutzen. Damit bilden nicht mehr die Formate, sondern nur die Kompatibilität der Modelle und der daraus resultierenden Architekturen der zusammenzuschaltenden Systeme die Grenzen der Austauschbarkeit einzelner Komponenten. Ein Beispiel für eine solche Inkompatibilität wäre z. B. die Reihenfolge von Dauer- und Intonationsmodulen in der prosodischen Prädiktion. Ein hypothetisches System A könnte für die Berechnung der Intonationskontur zwingend die Dauern der Segmente voraussetzen, während die Berechnung der Dauern in System B von der Ausprägung der intonatorischen Parameter abhängt. Solche Systeme können – zumindest ohne Abstriche in der Qualität – nicht ohne Weiteres verknüpft werden. Eine aus dem *TC-STAR*-Projekt hervorgegangene Erweiterung von SSML zu einer Repräsentationssprache präsentieren Pérez et al. (2006) für die Evaluationsaktivitäten im Rahmen des *European Center of Excellence for Speech Synthesis*, ECESS (2008). Die Hauptaufgabe des ECESS-XML ist ebenfalls die Zusammenschaltung verschiedener Systemkomponenten unterschiedlicher Provenienz, um deren Performanz systematisch vergleichen und evaluieren zu können. Diese Aufgabe ist allerdings dadurch vereinfacht, dass die Syntheseaufgabe in ECESS in lediglich drei große Aufgabenblöcke eingeteilt ist, hinter denen sich üblicherweise jeweils mehrere Syntheseschritte der zu evaluierenden Systeme verbergen. Damit existieren nur zwei systemübergreifende Schnittstellen in der ECESS-Architektur, nämlich zwischen den Blöcken *Textvorverarbeitung* (inkl. Transkription)/*Prosodie* einerseits und *Prosodie*/*akustische Synthese* andererseits. Auch das Institut für Kommunikationswissenschaften (IfK) beschäftigt sich im Rahmen der ECESS-Evaluation mit der Konvertierung von ECESS-XML in BOSS-Server-XML.

4.2.2. Aufbau des BOSS-Servers

Der BOSS-II-Server ist die Anwendung, die alle phonetischen und akustischen Verarbeitungsschritte der Synthese in sich vereint. Das Hauptprogramm ist dabei für das Einlesen der XML-Dokumente und die Ausgabe der Sprachsignaldaten zuständig. Es

instanziiert Objekte der Klassen für die Herstellung einer Verbindung zum Datenbankserver (`MySQL_Handler`) und für die Speicherung der Kommandozeilenoptionen (`BOSS_Commandline`). Der Kern der Synthese ist jedoch das Objekt `synthesis` vom Typ `BOSS_Synthesis`, das wiederum die Module zur Ausführung der einzelnen Syntheseschritte in einer geordneten Reihenfolge aufruft. Das `synthesis`-Objekt nennt Stöber (2002) daher auch *Module Scheduler*. In BOSS II waren die verwendeten Module anfänglich die von Stöber entwickelte Transkription, die Unit Selection und das Modul zum Laden und Verketten der Sprachbausteine.

4.2.3. Transkription

Die Transkription bestand in der ersten Version von BOSS II zunächst ausschließlich aus einer einfachen Datenbankabfrage: Der im Orth-Attribut eines `<WORD>`-Elements gespeicherte Text wird in einer Tabelle der gleichen MySQL-Datenbank gesucht, die auch die Annotationsdaten enthält. Die Tabelle, genannt `exception_lexicon`, verfügt über zwei Spalten mit jeweils der orthografischen und der phonetischen Schreibweise eines Wortes in jeder Zeile. Der zum Orth-Attribut passende Eintrag wird ausgelesen, an den Silbengrenzen in `<SYLLABLE>`-Knoten aufgeteilt und die Phone der Silbe werden anhand einer in der Tabelle `phoneme_table` gespeicherten Liste von Segmenttypen in einzelne `<PHONE>`-Elemente zerlegt. Die weitere Verarbeitung verläuft analog zur Umwandlung von BLF in Inventar-XML mit anschließender Ergänzung des phonetischen Kontextes durch `addcontext`. Das bedeutet, die `TKey`-Attribute aller Elementebenen erhalten die phonetische Transkription der jeweiligen Einheit und Phone und Silben zusätzlich ein Attribut `Stress`. Der Wert von Stress wird für alle lexikalisch akzentuierten Silben und die darin enthaltenen Vokale auf den Wert `1` (Primärakzent) oder `2` (Sekundärakzent) gesetzt, für alle anderen Einheiten auf `0`. Die Attribute `CLeft` und `CRight` geben die linken und rechten Nachbarphone der Elemente an.

Die Erstellung der benötigten Tabellen wird in Abbildung 4.1.2 (S. 70) durch die Schritte 7–8 dargestellt. Zunächst wird die Tabelle der Phon-Einheiten mit einem SQL-Skript erstellt. Dann wird das Aussprachelexikon durch Erzeugen einer leeren Tabelle und anschließenden Import des *Bonn Machine-Readable Pronunciation Dictionary*, BOMP (IfK, 2008a; Portele et al., 1995), angelegt.

In einer späteren Entwicklungsstufe von BOSS II fügte Stöber einen zweiten Transkriptionsschritt für im Lexikon nicht verzeichnete Wörter hinzu, der eine orthografische Repräsentation in seine Morphkomponenten zerlegen und auf die phonetischen Entsprechungen der Teilstrings abbilden kann. Die Datenbasis dieses Verfahrens ist die deutsche Morphem-Liste von Ortmann (1993). Bröggelwirth (Stöber und Bröggelwirth, 2000) erweiterte das Verfahren um morpho-phonologische Regeln, um

adäquate Oberflächenrepräsentationen der verketteten Morph-Transkriptionen zu generieren.

Eine später ergänzte dritte Stufe der Transkription, die Graphem-zu-Phonem-Abbildungen, Silbifizierungen und Akzentzuweisungen mit Hilfe von Entscheidungsbäumen vornimmt, wird unter den eigenen Arbeiten in Abschnitt 6.3.2.4 vorgestellt.

4.2.4. Unit Selection

Die Einheitenauswahl in BOSS funktioniert auch heute noch nach den gleichen Prinzipien und Algorithmen wie die Verbmobil-Synthese (s. 2.3.1). Nach einer Vorauswahl (*Pre-Selection*), die auch die zu verwendenden Einheitentypen für einen bestimmten Abschnitt der Zieläußerung bestimmt, werden Einheiten- und Transitionskosten für alle Kandidatenfolgen berechnet. Danach wird der optimale Pfad durch den Graphen der Einheitenfolgen ermittelt. Anders als die Verbmobil-Synthese nimmt BOSS jedoch nach der Vorauswahl kein Pruning der Kandidatenmenge auf der Basis von Einheitenkosten mehr vor.

Nicht alle Vorauswahl- und Kostenkriterien aus Verbmobil wurden jedoch für BOSS II übernommen, teils weil bestimmte Parameter fehlten, die in den der Synthese vorgeschalteten Schritten der Verbmobil-Architektur generiert wurden, teils wohl aufgrund zeitlicher Einschränkungen. In Ergänzung zu Verbmobil waren dafür in BOSS II auch Silben zulässige Einheiten für die Vorauswahl und den Kandidatengraphen.

In der Vorauswahl waren zunächst die folgenden SQL-Abfragen an die Datenbank für den Übergang zwischen den Einheitenebenen vorgesehen, die später mehrfach verändert wurden, bis durch eigene Arbeiten eine konfigurierbare Pre-Selection geschaffen wurde (s. Abschnitt 6.3.3). Die aktuelle, stark erweiterte Konfiguration für das Deutsche wird in Abschnitt 6.3.3 vorgestellt. Die hier gezeigte alte Version beschreibt eine Momentaufnahme, die sich auf die Darstellung von Klabbers und Stöber (2001) stützt, aber um einen rekonstruierten abschließenden Schritt ergänzt wurde.

```
1  -- Wort-Auswahl
2  SELECT * FROM <inventory>_word_data WHERE TKey="TKey" AND PInt="PInt";
3
4  -- Silben-Auswahl, 1. Stufe
5  SELECT * FROM <inventory>_syllable_data WHERE TReal="TKey"
6           AND CLeft="CLeft" AND CRight="CRight";
7
8  -- Silben-Auswahl, 2. Stufe
9  SELECT * FROM <inventory>_syllable_data WHERE TReal="TKey"
10          AND (CLeft="CLeft" OR CRight="CRight");
```

```
11
12 -- Phon-Auswahl, 1. Stufe
13 SELECT * FROM <inventory>_phoneme_data WHERE TReal="TKey"
14          AND CLeft="CLeft" AND CRight="CRight";
15
16 -- Phon-Auswahl, 2. Stufe
17 SELECT * FROM <inventory>_phoneme_data WHERE TReal="TKey"
18          AND (CLeft="CLeft" OR CRight="CRight");
19
20 -- Phon-Auswahl, 3. Stufe
21 SELECT * FROM <inventory>_phoneme_data WHERE TReal="TKey";
```

Listing 4.2: Auswahlstufen der Pre-Selection in BOSS II.

Auf der ersten Stufe (Zeile 2) wird die Datenbank nach einem Wort abgesucht, dessen Transkription und Phrasenposition mit dem Ziel übereinstimmen. Wird kein Ergebnis zurückgeliefert, wird die Suche auf der Silbenebene fortgesetzt. Gesucht werden zunächst Silben mit dem phonetischen Kontext der Zielsilben (Zeile 5), alternativ Silben mit wenigstens einseitig richtigem Kontext. Ist auch diese Suche ergebnislos, wird eine notfalls zweistufige Suche auf der Phon-Ebene analog zur Silbenauswahl vorgenommen. Sind alle diese Schritte erfolglos, wird versucht ein Phon zu finden, das zumindest das Kriterium der Lautidentität mit dem Ziel erfüllt.

Der BOSS II-Prototyp enthielt nur eine einzige (Transitions-)Kostenfunktion, die eine Bewertung des euklidischen Abstands der angrenzenden Mel-Cepstrum-Koeffizienten zweier Einheiten bereitstellte und faktisch die Kriterien der direkten Konsekutivität von Einheiten, der Kontextbeurteilung, der Energiedifferenz- und der F0-Distanzminimierung, sowie die Bewertung der Stimmhaftigkeitsunterschiede in der Verbmobilsynthese ablöste (s. Abschnitt 2.3.1). Als die prosodische Verarbeitung in BOSS durch Bröggelwirth um eine Dauerprädiktion ergänzt wurde (s. Abschnitt 5.2.1), konnte auch wieder eine Einheitenkostenfunktion zur Bewertung von Dauerunterschieden eingeführt werden. Zur Einschätzung der intonatorischen Kontinuität wurde von Stöber später ein auf der Grundfrequenzbestimmung in der Sprachsignalverarbeitungssoftware *ESPS* (2002) basierendes Werkzeug `addf0` zur Erweiterung des Inventar-XML und der Annotationsdatenbank um F0-Mittelwerte an den Einheitengrenzen entwickelt. Damit konnten auch die F0-Distanzen (s. Abschnitt 2.3.1) wieder als separater Kostenterm in die Transitionskostenberechnung eingehen.

Weitere Ergänzungen der Kostenfunktionen werden bei den eigenen Arbeiten in Abschnitt 5.3.3 dargestellt.

4.2.5. Signalloader

Das letzte Modul, das durch den Module Scheduler aufgerufen wird, ist der `BOSS_Signalloader`, der für das Laden und die Konkatenation der ausgewählten Synthesebausteine zuständig ist. Dazu muss er in der Annotationsdatenbank über die Nummer der jeweiligen Einheit und die Map-Tabellen (vgl. Abschnitt 4.1.3) die Zuordnung zum übergeordneten Satzeintrag in der Tabelle `sentence_data` herstellen. Dieser enthält den Namen der Datei, in der die Einheit enthalten ist. Listing 4.3 stellt die dazu benötigten SQL-Datenbankabfragen dar. Die Methode `BOSS_Signalloader::getSentenceFileName` definiert diese drei Abfragen für die verschiedenen Einheitenebenen. Ist die gesuchte Einheit ein Wort, wird das SQL-Statement in den Zeilen 2–4 verwendet. Wird eine Silbeneinheit gesucht, verwendet der Signalloader die in den Zeilen 7–10 definierte Suche. Für Phone muss die Zuordnung zum entsprechenden Satz über den Zugriff auf drei Map-Tables gefunden werden (Zeilen 13–17).

```
1  -- Wort-Einheiten
2  SELECT sentence_data.File FROM sentence_data, sentence_map
3  WHERE  sentence_map.WordNum = "word_num"
4  AND    sentence_data.Num = sentence_map.SentenceNum;
5
6  -- Silben-Einheiten
7  SELECT sentence_data.File FROM sentence_data, sentence_map, word_map
8  WHERE  word_map.SyllableNum = "syllable_num"
9  AND    sentence_data.Num = sentence_map.SentenceNum
10 AND    word_map.WordNum = sentence_map.WordNum;
11
12 -- Phon-Einheiten
13 SELECT sentence_data.File FROM sentence_data, sentence_map, word_map,
       syllable_map
14 WHERE  syllable_map.PhoneNum = "phone_num"
15 AND    sentence_data.Num = sentence_map.SentenceNum
16 AND    word_map.WordNum = sentence_map.WordNum
17 AND    syllable_map.SyllableNum = word_map.SyllableNum;
```

Listing 4.3: SQL-Queries zur Ermittlung der Herkunftsdatei einer Einheit.

Hat der `BOSS_Signalloader` den Dateinamen ermittelt, lädt er die entsprechende Audiodatei. Diese muss als PCM-kodierte Audiodatei mit 16-Bit-Quantisierung und 16 kHz Abtastrate vorliegen und darf keinen Datei-Header besitzen (*raw audio*). Die Einheit wird aus dem Signal herausgeschnitten und mit eventuell vorher bereits geladenen Einheiten ohne Glättung oder Manipulation, also „hart", konkateniert.

4.3. Fazit

Die in diesem Kapitel beschriebene BOSS-Architektur war in vielerlei Hinsicht zu rudimentär, um nur durch die Anpassung an ein bestimmtes Korpus eine vollwertige Syntheseanwendung darstellen zu können. Dies war auch nicht das erklärte Ziel von BOSS. Problematisch war jedoch, dass ein Konzept und auch bestimmte Voraussetzungen in der Architektur fehlten, um Ergänzungen aus den Arbeiten Dritter an BOSS wieder in die Basis des Systems einfließen zu lassen. So zeigten Klabbers und Stöber (2001) mit ihrer Anpassung von BOSS an das Niederländische zwar, dass die Architektur eine Erweiterung auf andere Sprachen stark vereinfachen kann. Die Rückportierung der von ihnen dafür entwickelten Algorithmen stellte sich jedoch als gleichbedeutend mit einer Neu-Implementierung heraus, weil z. T. strukturell inkompatible Design-Entscheidungen in der holländischen Adaption und der in der Zwischenzeit weiterentwickelten deutschen Version getroffen wurden. Um solche Probleme für die Zukunft zu vermeiden, wurden Konzepte für die Anpassung von BOSS an verschiedene Sprachen und Anwendungen erstellt und die dafür notwendigen Voraussetzungen in der Code-Basis implementiert. Diese Arbeiten, die BOSS zu einer vollwertigen multilingualen und multifunktionalen Synthese-Architektur erweitern, werden im folgenden Kapitel vorgestellt.

5. Ausbau der BOSS-Architektur: Multilingualität und Multifunktionalität

„Du hast so viele Leben, wie du Sprachen sprichst." *Aus Tschechien*

„Wer hohe Türme bauen will, muss lange beim Fundament verweilen."
Anton Bruckner

BOSS, obwohl konzeptuell durchaus für multilinguale und multifunktionale Anwendungen vorgesehen, war in der Version II aufgrund der Hartkodierung von Pfaden sowie sprach- und projektspezifischer Algorithmen im Quellcode schlecht für die verteilte und parallele Anpassung an verschiedene Sprachen und Aufgaben geeignet. Schon um BOSS-Projekte mit zwei verschiedenen Konfigurationen durchzuführen, war eine Verzweigung des Quelltextes in zwei Entwicklerbäume notwendig. Allgemeingültige Änderungen am BOSS-Code, die an einem der Bäume durchgeführt wurden, mussten per Hand mühselig mit der Parallelversion abgeglichen werden. War dazu noch die Einbindung spezieller Module im BOSS-Server nötig, wurde der manuelle Abgleich von Funktionen zwischen den beiden Versionen zu einer kaum ohne Fehler zu bewältigenden Aufgabe. Betroffen waren hiervon zunächst die beiden hausinternen Projekte PiAVIDA (Kindermann, 2005) und klickTel (Abresch und Breuer, 2004; Breuer und Abresch, 2003). Nur durch die ständige direkte Kommunikation und Zusammenarbeit zwischen den Entwicklern war es möglich, eine gemeinsame Basis des Quellcodes zu pflegen und Anpassungen auszutauschen. Als nach Abschluss dieser Projekte die Anpassung an das Polnische begonnen wurde, war es offensichtlich, dass für die weitere, nun internationale Zusammenarbeit an BOSS eine umfassende Flexibilisierung der Codebasis dringend nötig war. Im Verlauf dieses Kapitels sollen die vorgenommenen Änderungen nach inhaltlichen Kriterien gegliedert dargestellt werden, ohne dass dabei notwendigerweise die Chronologie der Implementierung widergespiegelt wird.

5.1. Konfiguration und Kommunikation

Der zwingendste Schritt war sicherlich zunächst die Ablösung des alten Konfigurationskonzeptes. Anstatt in einer Konfigurationsdatei wurde in BOSS II sowohl die Definition von Optionen (wie z.B. Dateipfaden) als auch von deren Standardwerten direkt im Quelltext vorgenommen. Diese Werte konnten zwar beim Programmstart der einzelnen BOSS-Komponenten über Kommandozeilenoptionen angepasst werden, allerdings schien dies auf Dauer keine sehr komfortable Lösung. Zudem fehlte ein zentraler Erfassungsort für alle verwendeten Einstellungen der verschiedenen Programme und Module, so dass der Überblick unnötig erschwert wurde. Da die Optionen für alle von der Synthese verwendeten Module im aufrufenden Programm, hier also dem BOSS-Server, festgelegt wurden, war für die Einbindung neuer, sprach- und modulspezifischer Optionen immer auch eine Anpassung des Server-Quelltextes notwendig.

5.1.1. Das Konfigurationskonzept

Um diese Probleme zu umgehen, wurde ein Format für Konfigurationsdateien erarbeitet, das die gesamte Funktionalität der bestehenden Kommandozeilenverarbeitung abbilden konnte, wie z.B. die Unterscheidung zwischen binären Optionen – *flags* – und Optionen mit Übergabewerten. Dies ermöglichte eine Implementierung der neuen Konfigurationslösung in Form der Klasse `BOSS::Config` (Holčík und Breuer, 2005) unter Beibehaltung der alten Objekt-Schnittstellen, wodurch im Quellcode der BOSS-Module und -Hilfsprogramme nur minimale Anpassungen vorgenommen werden mussten. Es lag nahe, als Basis für das Konfigurationsdateiformat XML zu verwenden, da zum einen die Kommunikationsstrukturen von BOSS vollständig auf XML aufbauen, zum anderen zunehmend XML als Metasprache für Konfigurationsdateien genutzt wird. Nichtsdestotrotz gab es zum Zeitpunkt der Implementierung kein standardisiertes, plattformübergreifendes XML-Vokabular, auf das hätte zurückgegriffen werden können. Daher wurde eine auf BOSS zugeschnittene Struktur von Elementen und Attributen definiert, die eine Konfiguration mehrerer Sprachen und Synthese-Inventare innerhalb einer Datei erlaubt. Die Klasse `BOSS::Config` wurde so konzipiert, dass zur Laufzeit über die Angabe des Inventarnamens an eine Klassenmethode der zugehörige Satz von Optionen aktiviert und so jederzeit zwischen den Konfigurationen verschiedener Korpora umgeschaltet werden kann.

Eine BOSS-Konfigurationsdatei dient nicht nur zur Voreinstellung der Optionswerte für alle BOSS-Programme und -Module, sondern ist auch der Ort, an dem die für eine Sprache oder Anwendung benötigten Optionen definiert und dokumentiert werden. Dies sorgt für maximale Flexibilität bei der Anpassung von BOSS, da in der

Konfiguration eingetragene Parameter sofort durch Aufruf in den sprachabhängigen Modulen genutzt werden können, ohne dass der sprachübergreifende Teil des BOSS-Quelltextes angepasst werden müsste. Zudem ist durch die Doppelfunktion der Konfigurationsdatei keine Meta-Beschreibung von Parametern in weiteren XML- oder Text-Dateien erforderlich. Nur einige wenige Parameter werden im Quellcode von `BOSS::Config` direkt definiert. Außer diesen existieren keine korpusübergreifenden Optionen.

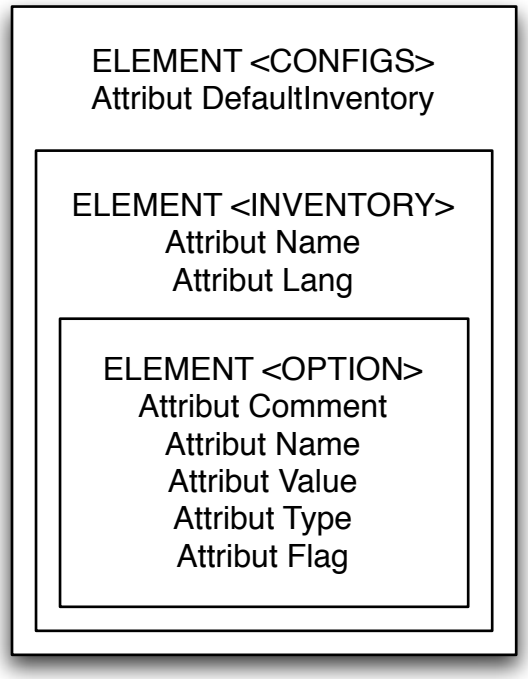

Abbildung 5.1.1.: Elementhierarchie einer BOSS-Konfigurationsdatei

Abbildung 5.1.1 zeigt den Aufbau einer BOSS-Konfigurationsdatei. Auf der obersten Stufe steht das Element `<CONFIGS>`, das alle anderen Elemente umschließt. Auf dieser Ebene lässt sich über das Attribut `DefaultInventory` ein Standard-Syntheseinventar aus den in der Konfigurationsdatei definierten Inventaren festlegen, das beim Start der Synthese verwendet wird, wenn keine Stimmauswahl per Kommandozeile getroffen wurde. Unterhalb der `<CONFIGS>`-Ebene können beliebig viele `<INVENTORY>`-Elemente angesiedelt sein, die wiederum beliebig viele `<OPTION>`-Elemente unter sich versammeln. Somit liegen alle für eine Synthesestimme relevanten Konfgurationsoptionen unterhalb eines `<INVENTORY>`-Knotens. Dessen Attribut `Name` enthält die Bezeichnung der Stimme, die von BOSS zur Auswahl der passenden Annotationsdaten aus der Datenbank benötigt wird. Über das Attribut `Lang` wird die Sprache des Inventars angegeben. Diese ist für die Auswahl geeigneter Kosten-

funktionen bei der Einheitenauswahl relevant (s. Abschnitt 5.3.3). Die Sprachbezeichnung für neu zu implementierende Sprachen ist frei wählbar, empfohlen wird aber ein standardkonformer zwei- oder dreistelliger Sprachkode nach ISO 639-1 oder -2 (ISO, 2002), um Ambiguitäten zu vermeiden. Für das Deutsche wird der Kode `de` verwendet.

Alle Optionen, die auch über die Kommandozeile verändert werden können sollen und nicht zur Auswahl des Inventars oder der Konfigurationsdatei dienen, werden über die Attribute der `<OPTION>`-Elemente gesetzt. Dazu gehören die Pfade zu weiteren, sprach- und inventarspezifischen Konfigurationsdateien, wie Symbollisten, Entscheidungsbäumen etc., aber auch Gewichtungsparameter für die Einheitenauswahl und Schalter für einzelne Funktionen der Synthesemodule. Die spezifizierten Attribute geben den Namen der Option an, der auf der Kommandozeile und im Quellcode verwendet wird (`Name`), den Vorgabewert der Option (`Value`), sowie einen Kommentar, der die Funktion des Parameters beschreibt (`Comment`). Da darüber hinaus auch reine Schalteroptionen (auch Binäroptionen/Flags) benötigt werden, die keinen sinnvollen Parameterwert besitzen, und die zum Start von BOSS entweder als gesetzt oder nicht gesetzt deklariert sein müssen, existieren zwei weitere Attribute. Diese legen fest, ob es sich um eine Option mit Argument oder einen Schalter handelt (Attribut `Type`) und ob der Schalter ggf. gesetzt ist (Attribut `Flag`). Eine Übersicht über die Verarbeitung verschiedener Attributwerte beim Aufruf von BOSS bietet Tabelle B.1.1 im Anhang. Ebenfalls im Anhang findet sich mit Listing B.1 ein Auszug aus einer XML-Konfigurationsdatei.

Um bei gleichzeitiger Erhaltung der Flexibilität einen besseren Abgleich zwischen den definierten und den im BOSS-Quelltext verwendeten Optionen zu schaffen, ist geplant, die Klasse `BOSS::Config` um eine Methode zu erweitern, mit der einzelne Synthesemodule Optionen anmelden können. Damit könnte schon während der Initialisierungsphase der Module – und nicht erst, wenn die Optionen zur Laufzeit zum Einsatz kommen – überprüft werden, ob ein benötigter Parameter fehlt und ggf. ein frühzeitiger Programmabbruch mit entsprechender Fehlermeldung vorgenommen werden.

Der bevorzugte Weg, zu synthetisierende Daten an den BOSS-Server zu übermitteln ist, das vom Server erwartete XML-Dokument durch eine Client-Anwendung über die Netzwerkschnittstelle zu senden. Zu diesem Zweck wurde bereits in einer frühen Version von BOSS II ein proprietäres, auf TCP/IP aufsetzendes Netzwerkprotokoll in Form der Klasse `BOSS_CSComm` implementiert, das eine XML-Datei und ein Sprachsignal senden und empfangen kann. (Stöber, 2002) stellt dieses Protokoll in seiner Dissertation vor. Zuvor gab es nur die Möglichkeit, die XML-Datei direkt beim Aufruf von BOSS über die Kommandozeile zu übergeben. In diesem Fall beendete sich der Server unmittelbar nach Synthese des XML-kodierten Eingabetextes und

schrieb die synthetische Äußerung über die Standardausgabe heraus. Diese Funktion wurde auch nach Implementierung der Netzwerkfunktionalität beibehalten.

Während der Arbeit am Telefonauskunftsprojekt (Kapitel 6) kam der Wunsch auf, über die Eingabe auch solche Metadaten zu einzelnen Äußerungsteilen übermitteln zu können, die nicht unmittelbar für die Synthese notwendig waren. Zum einen sollte ein Dateiname für die einzelnen Ausgabesätze definiert werden können. Damit sollten die Ausgaben der Testläufe in der Entwicklungsphase übersichtlicher gestaltet werden. Zum anderen war eine Forderung des Auftraggebers, dass auch unterhalb der XML-Notation, auf der Netzwerkprotokollebene, Indizes für die einzelnen Äußerungen gesendet und wieder empfangen werden können sollten. So sollte eine einfache Zuordnung zwischen Text und Signal ermöglicht werden, die ohne ein Zurücksenden der XML-Daten mit dem Signal und ohne einen XML-Parser[1] bewerkstelligt werden konnte.

5.1.2. Erweiterung des Netzwerkprotokolls

Um dem oben formulierten Anspruch gerecht zu werden, wurde das Netzwerkprotokoll von BOSS erweitert. In der alten Version des Netzwerkprotokolls sendet der Client zunächst eine Bitfolge fester Länge, die vom Server als Größe des folgenden XML-Dokumentes interpretiert wird. Der folgende Datenstrom wird genau bis zu dieser Länge eingelesen. Nach erfolgter Synthese wird zunächst die Größe des Signals gesendet. Der Client empfängt daraufhin dementsprechend viele Bytes.

Abbildung 5.1.2 zeigt den Ablauf des Netzwerkprotokolls. Die blau umrahmten Datenfelder wurden in der aktuellen Fassung hinzugefügt. Sie ermöglichen sowohl dem XML-Dokument als auch dem zurückgegebenen Signal einen Beschreibungstext hinzuzufügen. Im Normalfall ist dies die ID der Äußerung und für Ein- und Ausgabe gleich. Ihre Übermittlung erfolgt analog zu der des XML-Dokumentes und des Signals.

Um die plattformübergreifende Nutzung von BOSS voranzutreiben, wurde das Netzwerkprotokoll vom Autor in verschiedenen Programmiersprachen implementiert. So existiert eine Java-Klasse, die in einem für den im Rahmen eines Synthesepraktikums am IKP von Goldschmitz geschriebenen BOSS-Java-Client zum Einsatz kommt. Eine C++-Version, die mit den Borland-Bibliotheken unter Windows arbeitet, wurde für die Entwicklung des polnischen Syntheseclients von Szydłowski, Breuer und Szymański geschaffen. Von der älteren Protokollvariante existiert eine C#-Variante, die in einer .NET-Implementierung des `BOSSClient` von Weiß genutzt wird. Zudem wurde die erweiterte C++-Version für UNIX-Systeme compiler- und hard-

[1] – der im Client nicht standardmäßig benötigt wird –

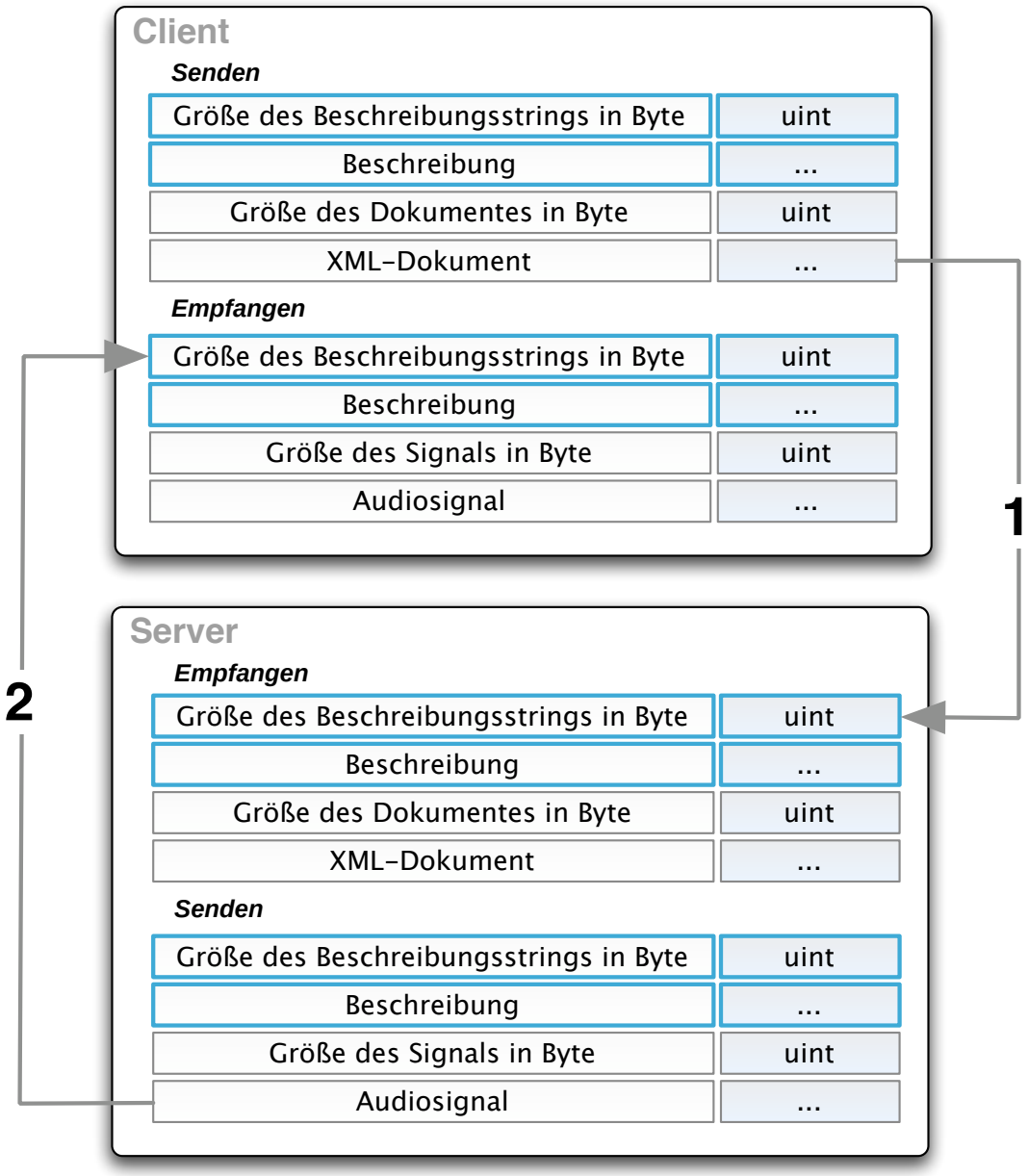

Abbildung 5.1.2.: Ablauf des Netzwerkprotokolls für den Austausch zwischen BOSS-Client und -Server. Neue Datenfelder sind durch einen blauen Rahmen gekennzeichnet.

wareunabhängig umgeschrieben, um Byte-Order-Probleme (Big-Endianness/Little-Endianness) zu umgehen und Unterschiede in der Länge gleichnamiger Datentypen auszugleichen. Dies war für die reibungslose Kommunikation zwischen heterogenen Clients und Servern unerlässlich.

5.1.3. Implementierung einer Dateischnittstelle

Für die Analyse synthetisierter Daten war es während der Entwicklung wichtig, einzelne Sätze als Datei ausgeben und sinnvoll benennen zu können. Dies mit den Mitteln der Shell zu bewerkstelligen hätte bedeutet, den BOSS-Server für jeden einzelnen Satz starten und beenden zu müssen. Weniger zeitaufwändig und eleganter wäre die Möglichkeit, auch längere Äußerungen an BOSS übergeben zu können, deren Metainformationen über die Satzelemente durch alle Module transportiert und schließlich für die Benennung von Ausgabedateien verwendet werden könnten. Um dies zu ermöglichen wurden dem Vokabular der BOSS-Eingabe-XML die in Abbildung 5.1.3 blau markierten Attribute hinzugefügt.

Die Attribute werden im BOSS-Server wie folgt verarbeitet, wenn er über die Option `-file` im Datei-Modus gestartet wird:

Der Wert von `TargetFile` wird als Namensbasis für alle Ausgabedateien (mindestens also die Audiosignale) genutzt. Daran wird, durch einen Unterstrich (_) separiert, der in `SentenceID` gespeicherte String gehängt. Ist `TargetPath` gesetzt, werden die Ausgabedateien am dort angegebenen Ort gespeichert. Sind diese Attribute nicht im Satzelement enthalten, werden sie vom BOSS-Server ergänzt. `TargetFile` und `TargetPath` entsprechen dann dem Herkunftsort und Namen der Eingabedatei, `SentenceID` erhält als Wert die laufende Nummer des Satzes in der Äußerung.

Auf dieser Basis war es nun möglich, den BOSS-Server und seine Module an jeder Stelle spezifische Informationen zu den verarbeiteten Sätzen herausschreiben zu lassen, die später wieder den Eingabesätzen zugeordnet werden konnten. So kann z. B. das Modul `BOSS_Concat` (s. Abschnitt 5.2.6) die ungeglätteten Signaldateien mit dem Zusatz `_plain.raw` speichern. Der Server selber erhielt Funktionen, um zu den Sätzen gehörige Label-Dateien im BLF-Format (Abschnitt 4.1.1) zu sichern, die verwendeten Parametereinstellungen sowie die aus der Gesamtheit aller Syntheseschritte resultierende maximale XML-Struktur in Dateien abzulegen.

5.2. Dynamisierung der Modulstruktur

Einer der Schritte, die für den Sprung hin zu einem multilingualen System am dringendsten erforderlich schienen, war die striktere Trennung von sprachabhängigem

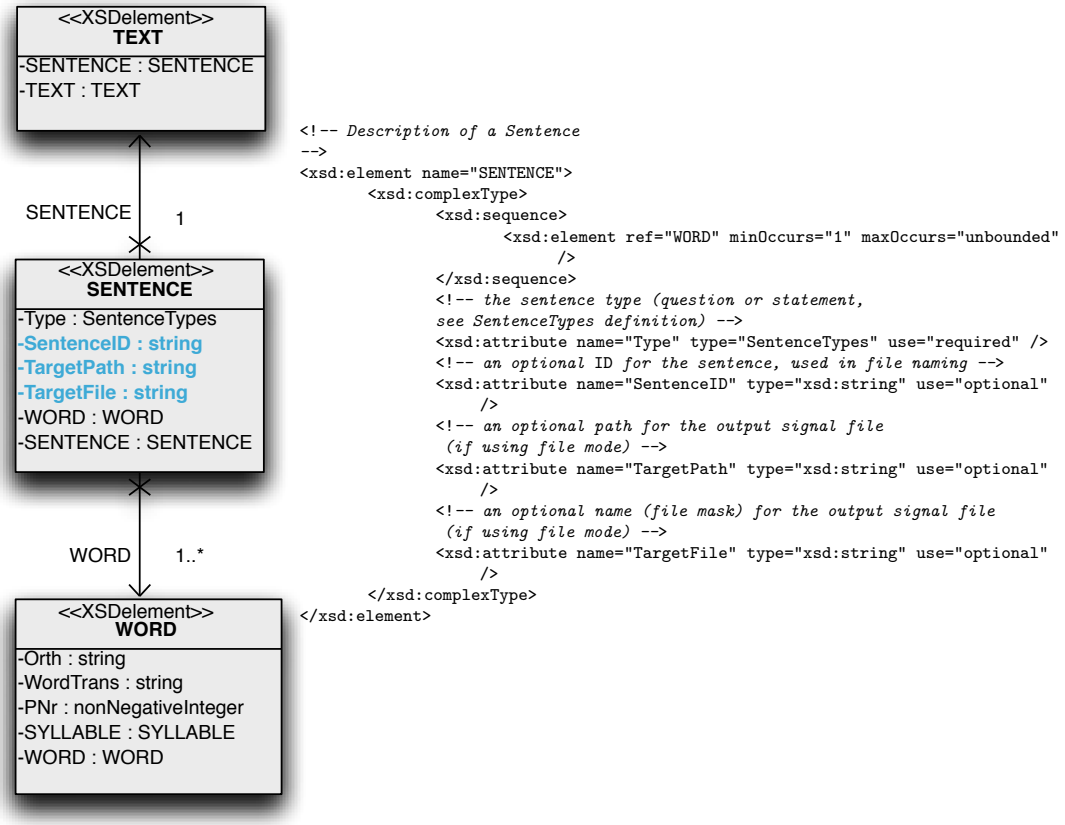

Abbildung 5.1.3.: Ausschnitt aus der deutschen XML-Schema-Definition eines BOSS-Server-Dokuments mit Fokus auf der Satzebene. Die neuen Attribute sind im Diagramm blau eingefärbt.

und sprachunabhängigem Code, die Wiederverwendbarkeit von Teilmodulen und die flexible Ein- und Ausschaltbarkeit einzelner Module ohne die Notwendigkeit, den Quellcode umzuschreiben und neu zu kompilieren. Ersteres war konzeptuell bereits in einem der ersten von Stöber entwickelten Symbolverarbeitungsmodule zur Transkription des Deutschen angelegt, musste aber noch konsequent ausgebaut und auf andere, später entstandene Module angewendet werden. Dabei ergab sich immer wieder das Problem der Wahl sinnvoller Schnittpunkte im Code. Als Beispiel soll die vom Autor durchgeführte Aufteilung des deutschen Dauervorhersagemoduls von Bröggelwirth dienen.

5.2.1. Funktionalität des deutschen Dauermoduls

Die Funktionalität des deutschen Dauermoduls umfasste ursprünglich die folgenden Aufgaben: Beim Initialisieren des Moduls wird zunächst ein zur Dauervorhersage

CART-Merkmal	BOSS-Attribut	Beschreibung
`LAUT`	TKey	Der Laut, für den die Vorhersage erstellt werden soll
`PLAUT`	CLeft	Der linke Nachbarlaut
`FLAUT`	CRight	Der rechte Nachbarlaut
`FLAUT2`	CRight2	Der übernächste Laut in Zeitrichtung
`PHRPOS`	Phrpos	Die Position in der Phrase (s. Formel 5.2.1)

Tabelle 5.2.1.: Die für das Deutsche verwendeten Dauervorhersageparameter. Die erste Spalte gibt den *feature*-Namen im CART an, die zweite das korrespondierende Attribut in BOSS-XML. In der dritten Spalte wird die Semantik des Parameters beschrieben.

trainierter CART-Entscheidungsbaum (*classification and regression trees*, Breiman et al., 1984) in dem durch die Software *wagon* (Black et al., 1999) erzeugten Format eingelesen, geparst und im Speicher abgelegt. Wird dem Modul zur Laufzeit der Synthese ein Satz zur Dauerprädiktion übergeben, durchläuft es alle darin enthaltenen DOM-Knoten vom Typ `<PHONE>` und reichert die Knoten jeweils mit den aus dem Satz ableitbaren aber auf der `<PHONE>`-Ebene noch nicht annotierten Daten an, die für die Prädiktion benötigt werden. Für die deutsche Vorhersage ist dies die Position des Lautes in der Phrase, ausgedrückt durch die Werte `I, M, F` (initial, medial, final) des Parameters `Phrpos`. Dieser Prädiktionsparameter wird, der allgemeinen BOSS-Konvention folgend, als Attribut des `<PHONE>`-Knotens bzw. -Elements gespeichert. Der Wert von `Phrpos` wird nach Formel 5.2.1 berechnet. Der Parameter n steht für die Anzahl der Silben in der Phrase, i ist der Index der übergeordneten Silbe des betrachteten Phons.

$$Phrpos = \begin{cases} \text{,,I”}, & \text{wenn } \frac{n-i}{n} > 0,5 \\ \text{,,F”}, & \text{wenn } i = n \\ \text{,,M”}, & \text{sonst} \end{cases} \tag{5.2.1}$$

Im nächsten Schritt durchläuft das Modul erneut alle Phone und überprüft anhand der Präsenz des Attributes `Dur`, ob die Dauern bereits in einem vorangehenden Modul erzeugt wurden und schon zur Verfügung stehen. Ist dies nicht der Fall, wird im Entscheidungsbaum ein zu den Parameterwerten des Lautes passender Pfad bis hin zum korrespondierenden Blatt gesucht. Der dort gefundene Wert stellt den Mittelwert der im Training gefundenen ähnlichen (merkmalsgleichen) Fälle in ms dar. Er wird als Dauervorgabe für die Einheitenauswahl im Attribut `Dur` des `<PHONE>`-Elements gespeichert. Tabelle 5.2.1 zeigt die für die Vorhersage verwendeten Merkmale.

Steht ein Laut in der letzten Silbe einer Phrase (`Phrpos = F`), wird der Dauerwert zusätzlich durch Formel 5.2.2 modifiziert, um den Effekt der phrasenfinalen Längung zu modellieren. Die Variable i steht hier für die Position des betreffenden Lautes in der Silbe, n für die Gesamtzahl der Laute in der Silbe. Mit dem Faktor x kann eine Korrektur des veränderten Wertes vorgenommen werden. In der Praxis wurde für das deutsche Lioba-Korpus ein Wert von $x = 0,65$ als günstig ermittelt.

$$dur = \exp\left(\frac{i}{n}\right) \cdot x \tag{5.2.2}$$

Auch in den Silben- und Wort-Knoten werden die aufsummierten Dauerwerte der diesen Einheiten untergeordneten Phone abgelegt. Damit sind die Dauern redundant auf allen Auswahlebenen vorhanden.

5.2.2. Überlegungen zur strukturellen Veränderung und Implementierung

Abbildung 5.2.1 stellt die Klassenstruktur der alten und neuen Implementierung der Dauervorhersage in einem UML-Diagramm (*unified modeling language*, Object Modeling Group, 2007) dar[2]. Die Klasse `BOSS_Duration` war in der Ursprungsfassung eine abstrakte (nicht-instanziierbare) Oberklasse für die abgeleitete Klasse `BOSS_Duration_DE`. Die Oberklasse bestand ausschließlich aus einer rein virtuellen (nicht implementierten) Methode zur Übergabe eines DOM-Knotens vom `<SENTENCE>`-Typ mit Hilfe des `()`-Operators. Diese Methode stellt die grundlegende Schnittstelle zum BOSS-System dar und muss in allen BOSS-Modulen enthalten sein. Damit bot `BOSS_Duration` keinerlei Funktionalität oder auch nur Schnittstellen, die für die Aufgabe der Dauerprädiktion spezifisch wären. Auf der anderen Seite enthielt die abgeleitete Klasse `BOSS_Duration_DE` alle Algorithmen, die zur Bearbeitung der Aufgabe notwendig sind, abgesehen von den grundlegenden Methoden zur XML-Verarbeitung, die in BOSS durchgängig aus anderen Bibliotheken importiert werden. Diese Aufteilung war weder im Sinne der objektorientierten Programmierung, noch bot sie dem Entwickler von neuen Sprachversionen Zeitersparnisse oder Erleichterungen bei der Implementierung. Um zu einer flexibleren Einteilung der Klassen und Methoden zu gelangen, musste der Quellcode zunächst hinsichtlich seiner funktionellen Einheiten analysiert werden. Zunächst wurden dabei die Methoden zum Einlesen des CART und der Durchwanderung des Baumes als spezifisch für die Anwendung der Dauermodellierung identifiziert (`read_tree()`,

[2]Aus Platzgründen werden die Parameter der Methoden und die den Klassen zugehörigen Datenstrukturen nicht aufgeführt.

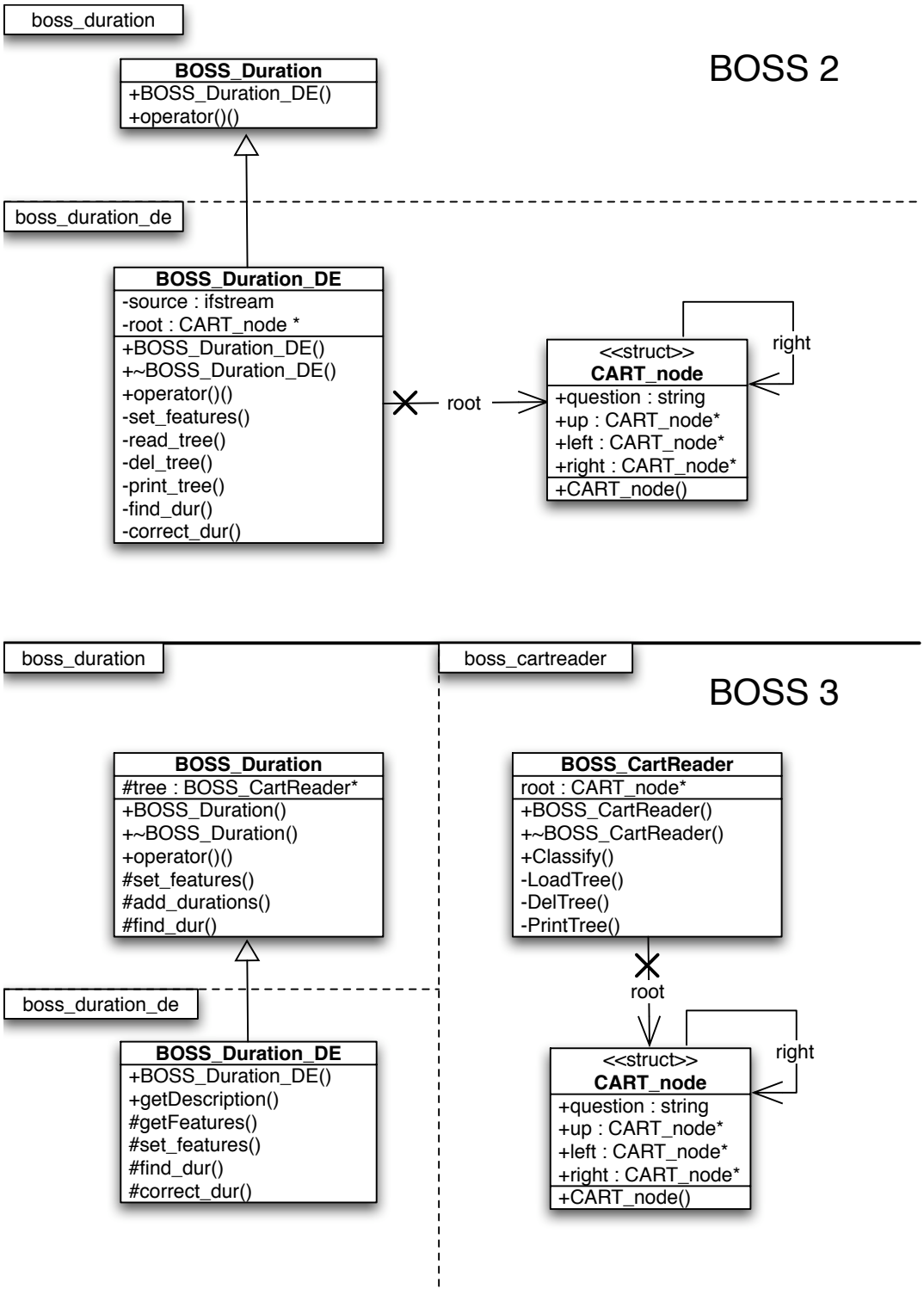

Abbildung 5.2.1.: Unterschiede in der Klassenaufteilung zwischen der ersten Implementierung der Dauerprädiktion und der aktuellen Version. Pfeile mit unausgefüllten Spitzen führen zu den jeweiligen Oberklassen. Durchkreuzte Pfeile zeigen auf innerhalb der Klassen implementierte C-*structs*.

`del_tree()`, `print_tree()`, `find_dur()`[2]). Aufgrund dieser Eigenschaft wären sie im sprachunspezifischen Teil der Anwendung, also der Oberklasse `BOSS_Duration` besser untergebracht. Allerdings sind CARTs ebenso gut auf andere Klassifikationsaufgaben anwendbar, wie z. B. die F0-Prädiktion. Um auch dem Prinzip der Wiederverwendbarkeit von Objekten gerecht zu werden, wurden die betreffenden Algorithmen und Datenstrukturen daher in eine eigenständige Klasse `BOSS_CartReader` und Bibliothek `bosscartreader` überführt und die Methoden mit allgemeingültigeren Namen versehen (`LoadTree()`, `DelTree()`, `PrintTree()`, `Classify()`[2]).

Die Funktion `set_features()` zur Ermittlung der Prädiktionsparameter war weniger eindeutig zu kategorisieren. Soweit die zu ermittelnden Parameter nur für die Dauerprädiktion nützlich waren, hätte man sie als spezifisch für diese Aufgabe betrachten und in die Oberklasse `BOSS_Duration` verschieben können. Es stellte sich allerdings die Frage, inwieweit die bisher verwendeten Parameter und deren Berechnung (s. Formel 5.2.1) universell auf andere Sprachen oder sogar andere Korpora derselben Sprache anwendbar wären. Da die Notwendigkeit einer vergleichbaren Vorverarbeitung für alle Sprachen und Korpora angenommen werden konnte, wurde entschieden, die Funktion als rein virtuelle Methode in die Oberklasse aufzunehmen und durch dieses Interface die Implementierung in abgeleiteten, sprach- oder korpusabhängigen Unterklassen verbindlich vorzuschreiben. Da die konkret verwendeten Parameter ursprünglich mit Hilfe der deutschsprachigen Bonner Prosodischen Datenbank (Heuft et al., 1995; IfK, 2008b) als die signifikantesten ermittelt wurden (Bröggelwirth, pers. Komm.) und im Anschluss mit vergleichbaren Ergebnissen für die Verbmobil-Datenbank eingesetzt wurden, konnten sie als hinreichend allgemeingültig angenommen werden, um in Verbindung mit CART zumindest für alle deutschen Korpora verwendet zu werden. Daher wurde die Implementierung in `BOSS_Duration_DE` belassen. In gleicher Weise wurde mit den Teilen der Methode `find_dur` verfahren, in denen die in Tabelle 5.2.1 dargestellten Abbildungen von BOSS-Attributen auf CART-Parameternamen vorgenommen und die Werte an den CART übergeben werden. Die eigentliche Traversion des Baumes befand sich ursprünglich ebenfalls dort, wurde aber konsequenterweise in der neuen Aufteilung Bestandteil der Methode `Classify()` von `BOSS_CartReader` (s. o.).

Mit der Methode `correct_dur()` wurde der in Formel 5.2.2 angegebene Korrekturterm implementiert. Damit sollte einem im Training auftretenden Mittelungseffekt zwischen den Dauern phrasenfinaler und phrasenmedialer Einheiten entgegengewirkt werden. Ob diese Korrektur für alle Sprachen oder Synthesekorpora notwendig ist, kann bezweifelt werden. Es wurde daher entschieden, die Funktion nicht in das Interface von `BOSS_Duration` aufzunehmen und sie in der Klasse `BOSS_Duration_DE` zu belassen. Um korpusabhängige Konfigurationen zuzulassen, sollen der Wert des bisher hartkodierten Parameters x und die Entscheidung, ob die Korrekturmethode

genutzt werden soll, in Zukunft über das Konfigurationsmodul einstellbar gemacht werden.

Die Speicherung und Propagierung des Dauerwertes in der DOM/XML-Struktur in Form des Attributes `Dur` war ursprünglich Teil der Aufrufmethode `operator()` von `BOSS_Duration_DE`. Diese Funktionalität wurde in eine Methode `add_durations()` ausgelagert, die bereits in der Basisklasse `BOSS_Duration` implementiert ist, denn die Weitergabe der ermittelten Dauerwerte ist spezifisch für die Aufgabe des Dauermoduls und in der vorliegenden Implementierung auch ausreichend allgemein gehalten, um auch für andere Sprachen und Korpora verwendbar zu sein. Die in der Basisklasse nicht implementierte `()`-Operator-Methode kann damit in den abgeleiteten Klassen flexibel ausgestaltet werden.

Natürlich ist, wie an allen Stellen, an denen die XML-Schnittstelle und deren Attribute sowie andere Interfaces berührt werden, auch diese Methode nicht komplett unabhängig von dem, was außerhalb des Dauer-Moduls in der umgebenden Anwendung implementiert ist. Sie baut beispielsweise darauf, dass eine bestimmte XML-Struktur mit den vorgegebenen Ebenen `<WORD>`, `<SYLLABLE>`, `<PHONE>`[3] existiert, in die die Werte eingetragen werden können[4]. Auch ergibt die Implementierung nur in einem Anwendungskontext Sinn, in dem die Dauerwerte genau aus dieser Struktur wieder extrahiert und auch verwendet werden, sei es zur Einheitenauswahl oder Dauermanipulation. Ähnliches trifft auch auf die sprachspezifischen Anteile in `BOSS_Duration_DE::find_dur()` zu. Die Ermittlung der CART-Eingabeparameter kann nur stattfinden, wenn die Attribute zum Lautkontext bereits im vorgeschalteten Transkriptionsmodul gesetzt wurden. Diese Anteile des Quellcodes und der Schnittstellen sollen hier anwendungsspezifisch genannt werden.

5.2.3. Systematisierung

Ausgehend von den Betrachtungen im vorangegangenen Abschnitt können wir eine Reihe von Kategorien ableiten, denen einzelne Programmelemente eines Synthesesystems angehören können. Obwohl am Beispiel des BOSS-Systems erklärt, sollte diese Systematisierung allgemein gültig und Ausgang jeder Designentscheidung beim Aufbau von Synthesearchitekturen sein können. Bisher wurden aufgabenspezifische, sprachspezifische, korpusspezifische und anwendungsspezifische Elemente identifiziert. Alle diese Elementkategorien stehen miteinander in Wechselwirkung und sind in der Praxis oft nicht vollkommen trennbar. Als Beispiel soll eine weitere, bisher

[3] Mittlerweile gehört dazu auch die Halbphon-Ebene, s. Abschnitt 5.3.4.

[4] Wobei angemerkt werden soll, dass die Implementierung fehlertolerant auf das Fehlen einzelner Ebenen reagiert

noch nicht eingeführte Kategorie dienen, die eng mit der Anwendungs- und Aufgabenspezifität in Zusammenhang steht - die der modellspezifischen Elemente. Hierunter sollen alle Codeanteile subsumiert werden, die mit der Implementierung eines bestimmten phonetischen Modells verknüpft sind, wie z. B. des Fujisaki-Modells (Fujisaki, 1988; Mixdorff, 1998; Möbius, 1993) für die Modellierung der F0-Kontur. Ein Intonationsmodul, das auf diesem Modell basiert, wird zu einem erheblichen Teil aus modellspezifischem Quelltext zur Verarbeitung und Berechnung der Parameter bestehen. Diese Anteile sind gleichzeitig auch spezifisch für die Aufgabe „Intonationsprädiktion". Andererseits impliziert die Entscheidung, das Fujisaki-Modell für diese Aufgabe zu verwenden, auch eine Designentscheidung für das gesamte System, da diese Parameter ja in einem späteren Modul auch verarbeitet werden müssen. Diese Elemente bedingen also anwendungsspezifische Code-Anteile in anderen Modulen und werden im Verbund mit diesen damit ebenfalls anwendungsspezifisch.[5]

Abbildung 5.2.2 stellt die Kategorien und ihre Wechselwirkungen dar.

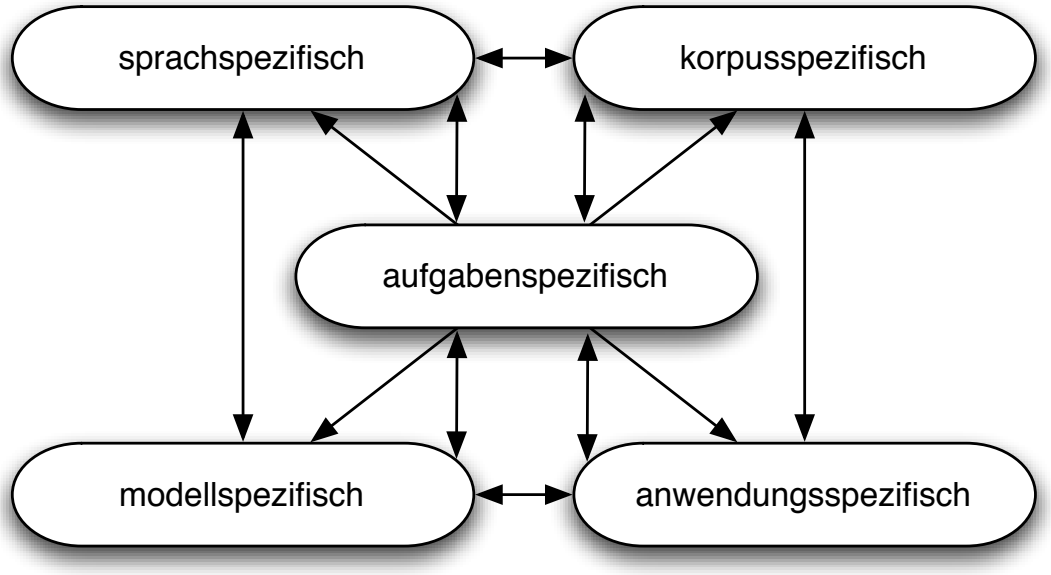

Abbildung 5.2.2.: Funktionale Komponenten einer Syntheseanwendung und ihre Wechselwirkungen

Zusammenfassend sollen die einzelnen Kategorien wie folgt definiert werden:

[5] Andererseits wäre auch denkbar, eine Interface-Schicht zu implementieren, die zwischen rein modellspezifischem Code und der Anwendung steht und die Ausgaben des Modells in eine andere Darstellung (z. B. mit Zeitmarken versehene F0-Werte) transformiert. Nur dieser Code wäre dann anwendungs-, aufgaben- und modellspezifisch. Ob eine solche Transformation ob des hohen Fehlers, der dadurch vermutlich induziert würde, sinnvoll ist, sei dahingestellt. Zumindest vom Bell-Labs-System (Sproat, 1998) ist bekannt, dass es teilweise mit solchen Umwandlungen arbeitete.

aufgabenspezifische Elemente sind solche Teile im Quelltext, die der Erfüllung einer bestimmten, i. d. R. standardmäßig benötigten Synthesefunktionalität dienen. Beispiele sind die automatische Transkription und die o. g. prosodischen Prädiktionsmodule;

sprachspezifische Elemente sind Werte und Algorithmen, die nicht für einen Großteil der natürlichen Sprachen sinnvoll verwendet werden können und/oder speziell nur für eine Sprache oder Sprachfamilie festgelegt und programmiert wurden;

korpusspezifische Elemente ähneln den sprachspezifischen Elementen, allerdings mit der Beschränkung auf nur einen Sprecher oder ein Korpus einer bestimmten Sprache. Auch Algorithmen und Parameter, die auf eine bestimmte Trainings- oder Datenmenge abgestimmt sind, etwa ein Trainingskorpus für die F0-Vorhersage oder ein Parametersatz für die artikulatorische oder formantbasierte Synthese können darunter fallen;

anwendungs- oder schnittstellenspezifische Elemente sind die Anteile des Programms, die eine Schnittstelle zu anderen Teilen der Synthese definieren, indem sie bestimmte syntheserelevante Daten *in einem festgelegten Format* von anderen Synthesebausteinen entgegennehmen oder für diese generieren;

modellspezifische Elemente sind solche Segmente im Programmtext, die ein bestimmtes für die Synthese relevantes phonetisches oder linguistisches Modell implementieren oder voraussetzen;

unspezifische Elemente sind Code-Anteile, die nicht ausschließlich für die Synthese genutzt werden, wie z. B. der `BOSS::CartReader` oder Bibliotheken von Drittanbietern.

Grob betrachtet sind die aufgaben-, anwendungs- und modellspezifischen Eigenschaften mit dem Konzept der Multifunktionalität verbunden, während der Begriff der Multilingualität sich auf die sprach- und korpusspezifischen Aspekte bezieht.

Anhand des vorgestellten Schemas können wir nun Regeln festlegen und Empfehlungen für das Design und die Programmierung zukünftiger BOSS-Module geben, die helfen sollen, die Ebenen möglichst vollständig zu trennen und damit eine größtmögliche Flexibilität des Codes zu gewährleisten.

5.2.4. Empfehlungen für die Strukturierung von BOSS-Modulen

Die Formulierung von Empfehlungen für die Weiterentwicklung von BOSS wirft eine grundsätzliche Frage zur angestrebten Funktion der gesamten BOSS-Architektur auf. Soll BOSS ein reines Framework für die Entwicklung von Unit-Selection-Systemen

bleiben und nur rudimentäre Schnittstellen zur Verfügung stellen, an die sprach- und anwendungsspezifische Module andocken können, oder soll es auch ein möglichst vollständiges und weitestgehend universelles *Out-of-the-box*-TTS-System darstellen? Die Antwort lautet, dass es zumindest dort, wo es praktikabel ist, auch Letzteres zu sein versuchen sollte. In diesem Geiste ist auch die Neukonzeption des Dauermoduls entstanden. Zwar stellt die Auslagerung universeller Funktionalitäten aus dem sprachspezifischen Modul einen Schritt hin zur kompletten TTS-Anwendung dar, doch ist ohne sprachspezifische Anpassungen keine Dauermodellierung für neue Sprachen mit dem (abstrakten) Basismodul möglich. Dies scheint sinnvoll, weil die Universalität der verwendeten Merkmale nicht geklärt ist. Auf der anderen Seite wäre es wünschenswert, dass bei der Konzeption neuer Module für andere Syntheseschritte versucht wird, dem Universalitätsanspruch möglichst weit entgegenzukommen. Konkret bedeutet dies, dass die Oberklassen für einen bestimmten Syntheseschritt (eine Aufgabe), soweit sinnvoll, nicht nur rein virtuelle Methoden als Schnittstellen definieren, sondern vollständige Implementierungen bieten. Diese Methoden sollten mit einem Satz von Ein- und Ausgabeparametern arbeiten, die als Mindestvoraussetzungen für alle Syntheseanwendungen als gegeben angenommen werden können und die minimal notwendig sind, um die Aufgabe zu erfüllen. Zusätzlich sollten sie aber *einfach* virtuell sein und damit von optionalen sprachspezifischen Unterklassen überschrieben werden können. Aus der Festlegung dieser Parameter ergibt sich durch die Abhängigkeiten der Module untereinander eine Gesamtheit von Syntheseschritten und deren Abfolge, sowie Schnittstellen in Form von Attributen und Werten der DOM/XML-Struktur. Das Ergebnis ist die Definition einer konkreten Anwendung - des *minimalen multilingualen BOSS-TTS-Systems*. Die im Folgenden gegebenen Empfehlungen werden sich, wenn nicht anders angegeben, auf diese Anwendung beziehen und der Begriff ,,anwendungsspezifisch" wird diese Verwendung der BOSS-Architektur bezeichnen.

Bei der Implementierung eines Syntheseschrittes ist die erste Entscheidung die für eine bestimmte Methode und/oder ein Modell, um die Aufgabe zu lösen. Hier können drei Fälle unterschieden werden, die mit unterschiedlichen Objektstrukturen des Moduls verknüpft sind:

1. Die verwendete Methode impliziert kein bestimmtes linguistisches Modell. Das Modell ergibt sich aus den sprach- und/oder korpusspezifischen Eingabeparametern und ist entweder über Konfigurationsdateien einstellbar oder in einer abgeleiteten Klasse definiert. Die Prädiktion wird von einem universellen Algorithmus durchgeführt. Beispiel: Das Modell der deutschen Dauerprädiktion besagt, dass Lautidentität, linker, erweiterter rechter Kontext, Wortakzent und Phrasenposition Korrelate der Lautdauer sind. Berechnet wird die Dauer durch den unspezifischen CART-Algorithmus.

2. Die verwendete Methode implementiert ein bestimmtes linguistisches Modell mit einem dedizierten Algorithmus für die Aufgabe. Die Parameter sind entweder hartkodiert und sprachuniversell oder die Implementierung erlaubt die Eingabe und Einstellung beim Aufruf durch sprachspezifische Module bzw. durch Konfigurationsdateien. Beispiel: das Campbell-Modell zur Dauerprädiktion auf Silbenbasis (Campbell, 1992).
3. Die verwendete Methode nutzt ein Metamodell, das zur Modellierung der Aufgabe, aber auch für andere linguistische Probleme verwendet werden kann. Die Implementierung erfolgt außerhalb der Modulbibliothek und wird damit wie unspezifischer Code behandelt. Beispiel: optimalitätstheoretische Algorithmen (OT, Prince und Smolensky, 2004).

Grundsätzlich gilt für die Programmierung von BOSS-Modulen, dass möglichst alle Sprach- und Korpusspezifika über Konfigurationsdateien abgedeckt werden sollten. Im Idealfall benötigt ein solches Modul keine sprachspezifischen abgeleiteten Klassen. Klassifikationsalgorithmen und andere unspezifische Hilfsfunktionen werden als eigenständige Klassen außerhalb des Moduls und der Modulbibliothek implementiert.

Abbildung 5.2.3 gibt Aufschluss darüber, wie die o. g. drei Fälle zu implementieren sind. Sie zeigt ein UML-Diagramm eines prototypischen BOSS-Moduls mit abstrakter mengentheoretischer Darstellung der empfohlenen Code-Aufteilung. Die beschriebenen Mengen stellen die spezifischen Code-Anteile gemäß dem im vorangegangenen Abschnitt vorgestellten Schema dar. Die Darstellung dieser theoretischen Modulstruktur orientiert sich an den folgenden Konventionen für die Benennung von Klassen, Bibliotheken und Verzeichnissen:

Die Oberklasse eines aufgabenspezifischen Moduls wird als Standard definiert und erhält den Gattungsnamen der Aufgabe, wenn die verwendete Methode oder das zugrundeliegende Modell universell einsetzbar sind oder als Grundlage für jede sprachspezifische Implementierung dienen können. Das minimale multilinguale BOSS-TTS-System verwendet beispielsweise die CART-Methode als Grundlage für die Dauerprädiktion aller Sprachen. Die Oberklasse, die spezifische Schnittstellen zwischen der Anwendung und CART implementiert darf daher (Beispiel: `BOSS_Duration`) heißen. Die abstrakte Schreibweise der Klasse lautet also: `BOSS_`*`Aufgabe`*. Sie wird in einem Unterverzeichnis von `boss/libraries` abgelegt. Verzeichnis und Dateinamen erhalten den Basisnamen `boss_`*`aufgabe`*. Die dazugehörige Bibliothek heißt `bossaufgabe`. Abgeleitete sprachspezifische Implementierungen erhalten den Zusatz *_ISO* (bzw._*iso*,*iso*: Platzhalter für den ISO-Sprachkode, s. Abschnitt 5.1.1), davon abgeleitete korpusspezifische Implementierungen den Zusatz *_Stimme* (_*stimme*, *stimme*). Beide werden in separaten Quelltextbäumen abgelegt, zusammen mit den

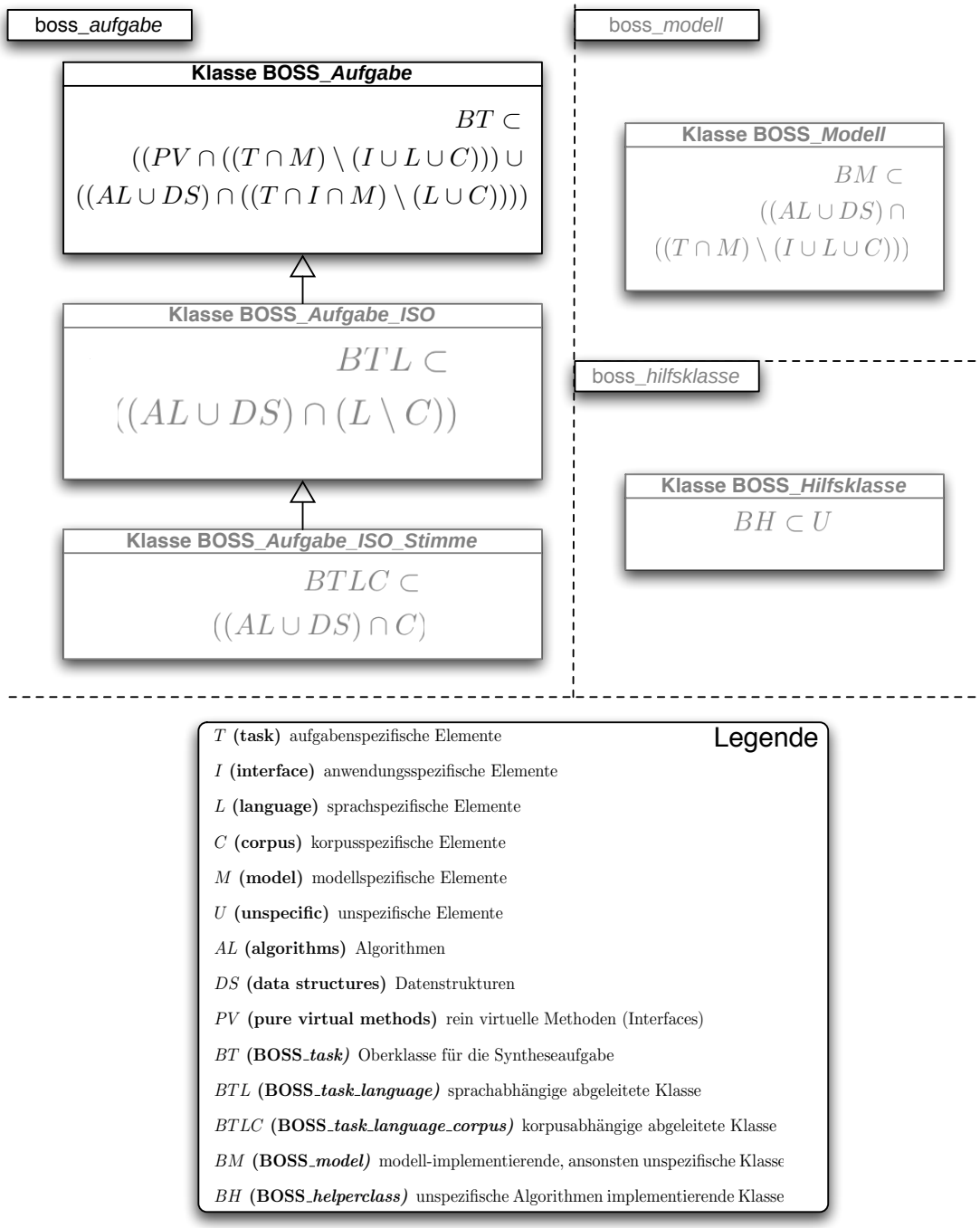

Abbildung 5.2.3.: UML-Diagramm eines prototypischen BOSS-Moduls mit abstrakter mengentheoretischer Darstellung der Code-Aufteilung. Optionale Elemente sind grau dargestellt.

sprach- oder korpusspezifischen Dateien der anderen Synthesemodule[6].

Eine Klasse BOSS_*Aufgabe* soll, wie in Abbildung 5.2.3 gezeigt, idealerweise nur solche Algorithmen und Datenstrukturen enthalten, die gleichzeitig modell-, und aufgabenspezifisch sind und das Interface zur Anwendung definieren, aber nicht sprach- und korpusspezifisch genannt werden können. Sie darf außerdem rein virtuelle Methoden definieren, wenn die Modellspezifika, wie im o. g. Fall 1 durch abgeleitete sprach- und korpusspezifische Klassen implementiert werden. Bevorzugt sollten aber, wie oben geschildert, möglichst sprachuniverselle Methoden implementiert werden, die ggf. überschrieben werden können.

Von der Oberklasse abgeleitete Klassen (BOSS_*Aufgabe_ISO*) enthalten alle sprachspezifischen Informationen zur Durchführung der Aufgabe, wenn nötig auch solche, die anwendungs- und modell-, aber nicht korpusspezifisch sind. Letztere werden wiederum in einer von den sprachspezifischen Anteilen abgeleiteten Klasse implementiert.

Wenn, wie in Fall 2 dargestellt, ein linguistisches Modell dedizierte Algorithmen für die Lösung einer Aufgabe erfordert, sollten diese in einer separaten Klasse nach der Schablone BOSS_*Modell* implementiert werden. Unter Umständen werden die korpus- und sprachspezifischen Klassen in diesen Fällen nicht benötigt. Eine Klasse BOSS_*Modell* enthält nur modell- und aufgabenbezogene Algorithmen und Datenstrukturen, die aber nicht vom Interface des multilingualen BOSS-TTS-Systems abhängen und nicht sprach- oder korpusspezifisch sein dürfen.

Metamodelle (Fall 3) werden im Sinne des Schemas als unspezifisch betrachtet und in der Implementierung wie andere Algorithmen behandelt, die nicht nur einer bestimmten Syntheseaufgabe dienen können, selbst wenn sie im Unterschied zu bspw. CART rein linguistische Modelle sind. Der Name solcher Klassen leitet sich aus dem Modellnamen ab. In der Abbildung werden sie durch BOSS_*Hilfsklasse* repräsentiert.

Natürlich handelt es sich hier um ein idealisiertes Schema, das in der Praxis aufgrund der dargestellten Interaktionen zwischen den funktionalen Komponenten (s. Abb. 5.2.2) nicht in allen Fällen umzusetzen sein wird. Nichtsdestotrotz fördern diese Richtlinien das Nachdenken über die Aufteilung von Syntheseaufgaben und sind so ein wichtiger Schritt zur Einheitlichkeit von Modulimplementierungen.

[6]Eine Ausnahme bilden die deutschen Module, die momentan als Referenz mit dem BOSS-Quelltext verteilt werden.

5.2.5. Flexibilisierung der Moduleinbindung

Mit der Einführung des Konfigurationskonzeptes (Abschnitt 5.1.1) und der Festlegung eines Schemas für die Aufteilung der Module hatte BOSS nun die Voraussetzungen, um an verschiedene Sprachen, Korpora und Anwendungen angepasst werden zu können. Für eine Entwicklung hin zu einer echten Multifunktionalität und Multilingualität, in dem Sinne, dass mehrere Anwendungstypen und Sprachen parallel mit dem gleichen BOSS-Server betrieben werden können, bedurfte es noch einer Möglichkeit, erst zur Laufzeit einzelne BOSS-Module einbinden und sprachabhängig zwischen verschiedenen Sets von Modulen umschalten zu können. In der Ursprungsversion von BOSS II war es erforderlich gewesen, die benötigten BOSS-Module explizit im Programmtext aufzurufen. Damit war es unumgänglich, mindestens diese Klasse analog zur deutschen Version neu zu implementieren, wollte man Module anderer Sprachen verwenden. Die resultierende Aufteilung grundlegender BOSS-Bausteine in verschiedene Entwicklerbäume war ein großes Hindernis für die Zusammenarbeit, und es wurden Überlegungen angestellt, wie die Synthese-Klasse flexibler gestaltet werden könnte.

Ein weiteres Problem der bestehenden Architektur war, dass zwar durch die Klasse `BOSS::Config` nun Konfigurationsoptionen für mehrere Korpora geladen werden konnten, die Auswahl des zu verwendenden Korpus jedoch immer noch hartkodiert war. Auch gab es nur eine Instanz der Klasse `MySQL_Handler`, die die Anbindung an die Annotationsdatenbank eines Korpus herstellt. Multilinguale Synthese mit der Möglichkeit, zur Laufzeit zwischen Sprachen zu wechseln, war somit ausgeschlossen. Ein neues Scheduler-Konzept (vgl. Abschnitt 4.2.2) musste geschaffen und an das Konfigurationskonzept angebunden werden. Es wurde daher eine Neuimplementierung der Klasse `BOSS_Synthesis` beschlossen, die folgende Kriterien erfüllen sollte:

- Die Anpassung an andere Sprachen oder Anwendungstypen soll ohne Veränderung des BOSS-API möglich sein;
- mehrere Gruppen von sprach- oder anwendungsspezifischen Modulen sollen parallel geladen sein können und
- zwischen diesen Gruppen soll abhängig von der Sprache des Eingabetextes satzweise umgeschaltet werden können[7];
- alle Gruppen müssen Zugriff auf ihre spezifischen Konfigurationsdaten haben und eigene Datenbankanbindungen besitzen.

[7]Eine vollständig polyglotte Anwendung mit einer wortweisen Umschaltung wäre realisierbar, indem die Satzgrenzen entsprechend gesetzt werden. Ob eine vollständige Umschaltung von einer vollständig nativen Aussprache einer Sprache in die einer anderen innerhalb einer Äußerung jedoch sinnvoll ist, ist spätestens seit der Dissertation von Abresch (2007) zur Aussprache des Englischen in deutschen Kontexten fraglich.

Die Grundvoraussetzung für eine Anpassbarkeit ohne Quelltextveränderung war, dass ein Mechanismus gefunden würde, der es erlaubte, unbekannte BOSS-Module zur Laufzeit einzubinden. Dies würde nur gelingen, wenn alle BOSS-Module die selben grundlegenden Schnittstellen zum Aufruf und zur Verwendung verwendeten. Auch wenn dies in der Ursprungsversion von BOSS II bereits so angelegt war, für die Verwendung im o. g. Sinne war es nötig, dass alle BOSS-Module nach dem Prinzip der Polymorphie (Cardelli und Wegner, 1985) von derselben Oberklasse abgeleitet werden und die gleiche Signatur besitzen. Daher implementierte Holčík (2005) eine abstrakte Klasse `BOSS::Module`, die lediglich die folgenden Interface-Definitionen, Methoden und Datentypen anbietet:

1. den Konstruktor, in dem eine Referenz auf die systemweite Instanz von `BOSS::Config` und ein Zeiger auf die korpusspezifische Instanz der Datenbankanbindung übergeben werden;
2. Attribute zur Speicherung der genannten Referenzen und Zeiger;
3. die allgemeine BOSS-Schnittstelle zur Übergabe eines Satzes (rein virtuell);
4. eine Schnittstelle zur Ausgabe einer Modulbeschreibung (rein virtuell);
5. Typdefinitionen, die zum dynamischen Laden erforderlich sind.

Aus allen BOSS-Modulen wurden die nun redundanten entsprechenden Quelltextelemente entfernt, und die Module wurden als abgeleitete Klassen von `BOSS::Module` definiert. Komplementär zu den Typdefinitionen implementierte Holčík eine Funktion (Breuer und Holčík, 2005), die auf Basis der Bibliothek `libltdl` (The Free Software Foundation, 2008) das Laden von `BOSS::Module`-Instanzen ermöglicht. Die vom Autor neu geschaffene `BOSS_Synthesis`-Klasse nutzt diese Funktion und bindet sie in das neue Scheduler- und Konfigurationskonzept ein. Der Ablauf eines Synthesevorganges mit Fokus auf `BOSS_Synthesis` wird durch ein UML-Sequenzdiagramm in Abbildung 5.2.4 gezeigt.

Er beinhaltet die folgenden Schritte, deren Nummerierung den *Message*-Indizes an den Pfeilen des Diagramms entspricht:

1. Die BOSS-Server-Applikation ruft den Konstruktor der `BOSS::Config-Klasse` auf, die wiederum die BOSS-Optionsdatei einliest;
2. `boss_server` instanziiert ein `BOSS_Synthesis`-Objekt mit dem Namen `synthesis` und übergibt ihm einen Zeiger auf das `Config`-Objekt;
3. `synthesis` erzeugt die Klassenvariable `invmap`, die später die Beziehung zwischen einem Inventarnamen und der zugehörigen Datenbankanbindung sowie dem Modulablaufplaner (Scheduler) herstellen wird;

Für jedes Inventar, das BOSS in der Optionsdatei findet, werden vom `synthesis`-Konstruktor die folgenden Schritte durchgeführt:

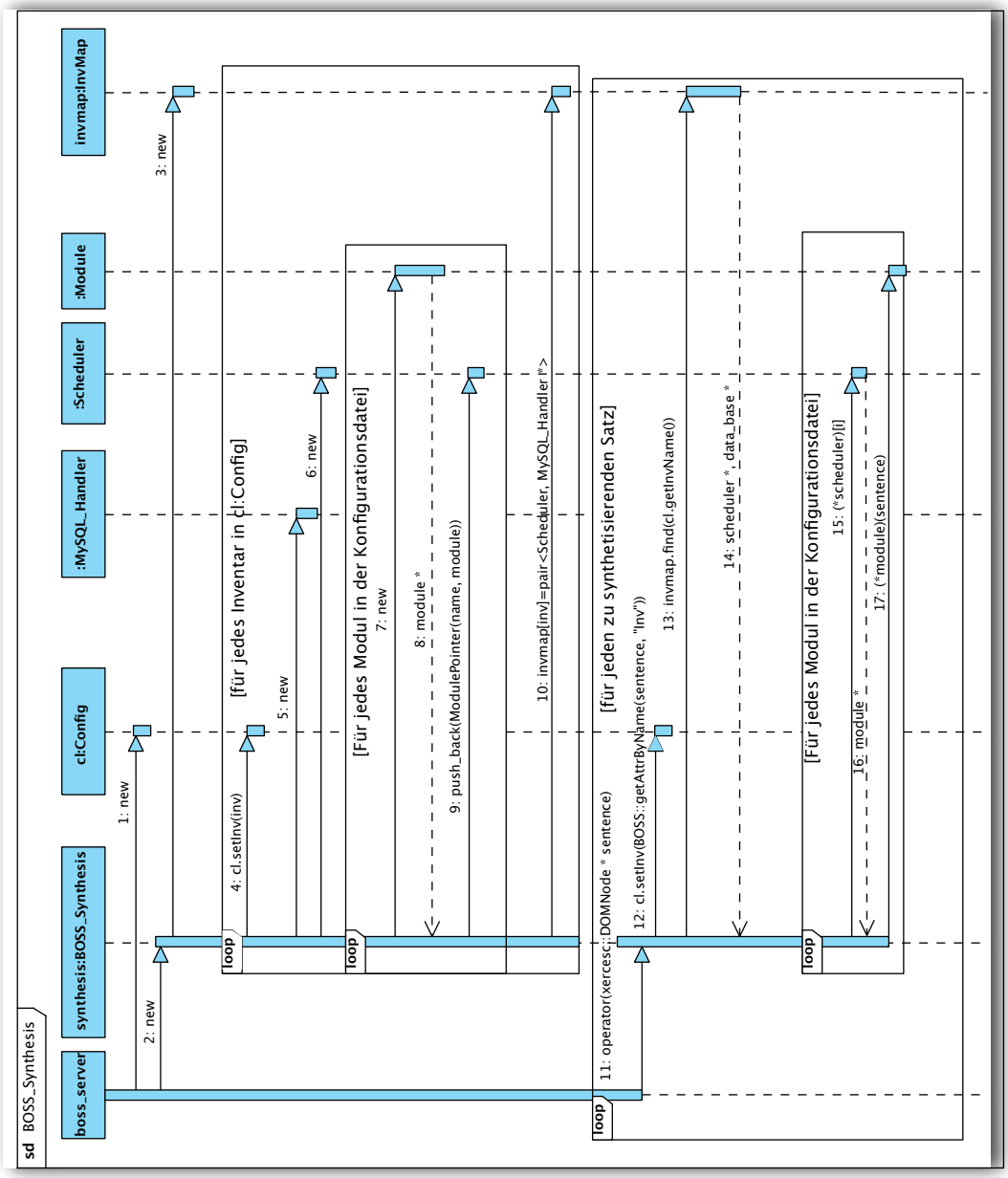

Abbildung 5.2.4.: UML-2-Sequenzdiagramm eines Synthesevorganges.

4. Der aktive Satz von Optionen in `Config` wird auf das aktuell betrachtete Inventar geschaltet;
5. die für das Inventar angegebene Datenbankverbindung wird hergestellt, indem ein `MySQL_Handler`-Objekt mit den entsprechenden Parametern erstellt wird;
6. für das Inventar wird ein `Scheduler`-Objekt instanziiert, das Zeiger auf BOSS-Module in einer festgelegten Reihenfolge speichert.

Innerhalb der Schleife wird eine weitere Schleife gestartet. Aus einer durch die Option ,,`-modules`" spezifizierten Konfigurationsdatei wird eine Abfolge von Bibliotheksnamen verschiedener BOSS-Module eingelesen, z. B.

`boss_transcription_de,` das deutsche Transkriptionsmodul;

`boss_duration_de,` das deutsche Dauermodul;

`boss_unitselection,` die Einheitenauswahl;

`boss_concat,` das Modul zur akustischen Synthese.

Für jeden dieser Bibliotheksnamen wird die zugehörige *Library* geladen und

7. das entsprechende Modul instanziiert und
8. der Zeiger auf die Instanz sowie
9. der Name an den Scheduler angehängt.

Die innere Schleife endet und

10. der Scheduler wird zusammen mit dem Datenbankzeiger in `invmap` gespeichert. Beide sind jetzt per Übergabe des Inventarnamens an `invmap` ansprechbar.

Die äußere Schleife und der `synthesis`-Konstruktor enden. Wird nun ein BOSS-XML-Dokument zur Synthese an den BOSS-Server gesendet, so wird dieses

11. Satz für Satz an den `synthesis-()`-Operator übergeben;
12. die Attribute des Satzes werden nach der Angabe eines Inventarnamens durchsucht und die aktive Konfiguration bei erfolgreicher Suche entsprechend umgestellt; ansonsten wird das Standard-Inventar aktiviert;
13. der Scheduler für das aktive Inventar und das Datenbankobjekt werden in `invmap` gesucht und
14. die von `invmap` zurückgegebenen Zeiger zur Verwendung mit dem aktuellen Satz gespeichert.

Der Reihe nach werden nun die Zeiger für alle Module aus dem Scheduler

15. extrahiert,
16. in einer Zeigervariablen gespeichert und

17. die Module aufgerufen, um den aktuellen Satz zu bearbeiten.

Nach Beendigung aller Schleifen ist der Ablauf von der Instanziierung des `BOSS_Synthesis`-Objektes bis zur Synthese eines BOSS-XML-Eingabedokumentes abgeschlossen. Die erstellten Objekte existieren bis zur Löschung von `synthesis` bzw. bis zum Abbruch der Server-Applikation weiter.

5.2.6. Struktur der Manipulationsmodule

Anders als die symbolverarbeitenden Anteile von BOSS sollten Module zur akustischen Verkettung und Manipulation möglichst vollständig sprachen- und korpusunabhängig sein. Vielmehr entscheidet die Art der Anwendung, welche Konkatenations- oder Manipulationsaufgaben benötigt werden. Strukturell ist also eine etwas andere Konzeption notwendig.

Ursprünglich war in BOSS keinerlei Manipulation der Sprachsignale nach der Auswahl und Zusammensetzung vorgesehen. Die von der Unit Selection ausgewählten Einheiten wurden über einen Satz von SQL-Anweisungen an das MySQL-Datenbanksystem der originären Sprachsignaldatei zugeordnet. Aus den Dateien wurden die zu verwendenden Segmente geladen und ohne jegliche Glättung oder andere Signalmanipulationen „hart" konkateniert. Diese Funktionen übernahm in der anfänglichen Version von BOSS II die Klasse `BOSS_Signalloader`, die damit auch das letzte Glied in der Kette der Syntheseschritte darstellte.

Zwar ist die geschilderte Vorgehensweise für die Unit-Selection-Synthese nicht ungewöhnlich, denn ob sich die durch Manipulation des Sprachsignals eingeführten Qualitäts- und Natürlichkeitseinbußen nicht möglicherweise schwerer auswirken als die ein oder andere prosodische oder spektrale Unstimmigkeit, ist eine offene Frage. Jedoch ist zumindest eine einfache Glättung an den Einheitengrenzen oder auch eine Optimierung des Konkatenationspunktes (z. B. Conkie und Isard, 1996; Prudon, 2002) wohl in den meisten Systemen vorhanden. Andererseits war der initial geäußerte Anspruch an die BOSS-Architektur, auch für die traditionelle konkatenative Synthese geeignet zu sein. Ohne eine Möglichkeit der prosodischen Manipulation wäre dies jedoch ein Rückschritt gegenüber den Vorgängersystemen. Weil auch im Rahmen des klickTel-Projektes (s. Kapitel 6) der Wunsch nach veränderlichen Dauerverhältnissen aufkam, wurde die alte `BOSS_Signalloader`-Klasse abgelöst und zunächst durch ein Modul mit dem Namen `BOSS_ConMan`[8] ersetzt. Anfänglich enthielt dieses sowohl die Funktionen des `Signalloader`s als auch rudimentären Code

[8]Diese Kurzform, die sich aus ***Con****catenation and* ***Man****ipulation* zusammensetzt, weist für den Englisch sprechenden Benutzer in doppelter Bedeutung auf die Hauptfunktion eines Manipulationsmoduls hin.

für die Dauermanipulation durch Aufruf eines Scripts. Mit diesem wurde die Verarbeitung an die PSOLA-Funktionen der Signalverarbeitungssoftware *Praat* (Boersma und Weenink, 2001) übergeben. Alternativ dazu konnte auch eine Glättungsfunktion aufgerufen werden, die an den Segmentengrenzen folgende, bei Prudon (2002) beschriebene Operation durchführte:

- Für jede Einheit wird zusätzlich jeweils ein kurzer Signalabschnitt links und rechts der Einheitengrenze geladen.
- Über die linken Signalabschnitte wird die erste Hälfte eines Hanning-Fensters gelegt, über die rechte Seite die zweite Hälfte.
- Bei der Konkatenation zweier Einheiten überschreitet die rechte Fensterhälfte der ersten Einheit die linke Einheitengrenze des Nachfolgers und umgekehrt. An den Überlappungsstellen werden die Werte aufaddiert.

Abbildung 5.2.5 illustriert das Verfahren für die Verkettung der Diphone _ʃ und ʃa.

Dass dieses einfache Verfahren Wirkung zeigt, lässt sich in Abbildung 5.2.6 erkennen. Insbesondere im spektralen Bereich sieht man, dass das Entstehen neuer Frequenzen deutlich vermindert werden konnte.

Als mit der Bachelor-Arbeit von Holčík (nicht veröffentlicht) eine eigene OLA-Implementierung für BOSS geschaffen wurde und die Arbeiten an einem *Harmonics-plus-Noise*-Modell (Rohde und Breuer, 2005) begannen, stellte sich die Frage, wie diese in das `ConMan`-Modul integriert werden könnten. Es wurde beschlossen, analog zu der Aufteilung der Symbolverarbeitungs-Module (s. Abschnitt 5.2.4) eine abstrakte Oberklasse zu schaffen, die möglichst allgemeingültigen Code bzw. lediglich Schnittstellen bereitstellen sollte.

Abbildung 5.2.7 zeigt die neue Aufteilung der oben geschilderten Funktionen Dateiermittlung, Einheitenextraktion und Glättung/Konkatenation. Im Sinne der in Abschnitt 5.2.3 vorgestellten Klassifikation sollen die ersten beiden Funktionen zusammen mit der Konkatenation und Manipulation aufgabenspezifisch genannt werden[9]. Das konkrete Verfahren der Manipulation (inkl. Glättung) wäre dann anwendungs- und modellspezifisch. Sprach- und korpusspezifische Anteile sind auf dieser Ebene zunächst nicht vorgesehen.

Aus der ursprünglichen Bibliothek `bossconman` wurden zwei Bibliotheken `bossconman` und `bossconcat`. `BOSS_ConMan` ist nunmehr eine abstrakte Klasse, die außer den Schnittstellen nur eine Methode zur Speicherung des erzeugten Sprachsignals in der DOM-Struktur enthält. Dadurch ist keine gesonderte Schnittstelle für die

[9] Andererseits ist die Funktionalität der Manipulation natürlich nur in einem konkatenativen System sinnvoll. Alle derartigen Module sind deshalb schon in ihrer Gesamtheit — auch weil sie auf die Schnittstellen für die minimale multilinguale Synthese zurückgreifen — anwendungsspezifisch.

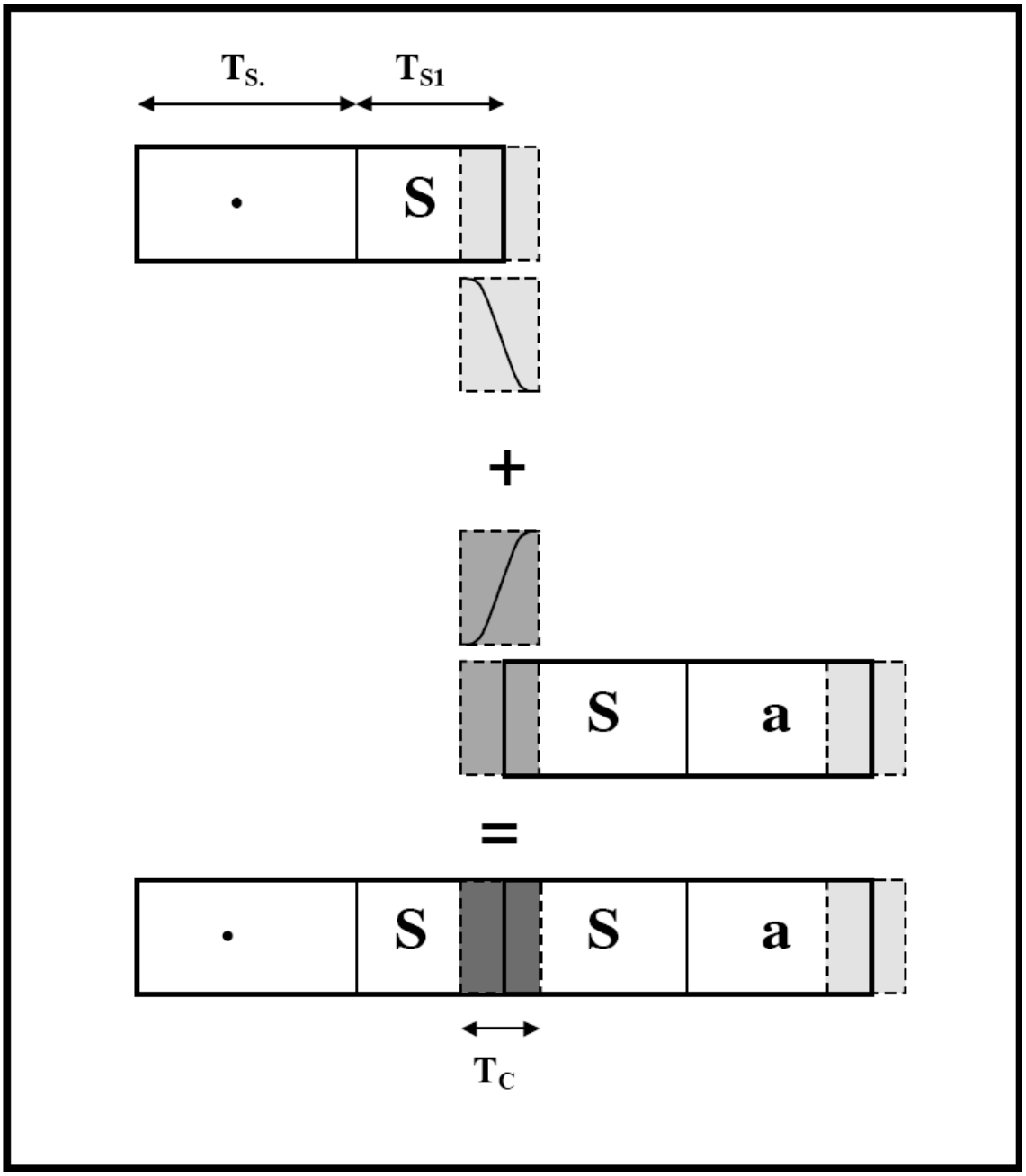

Abbildung 5.2.5.: Einfaches Glättungsverfahren für Verkettungsstellen (Prudon, 2002).

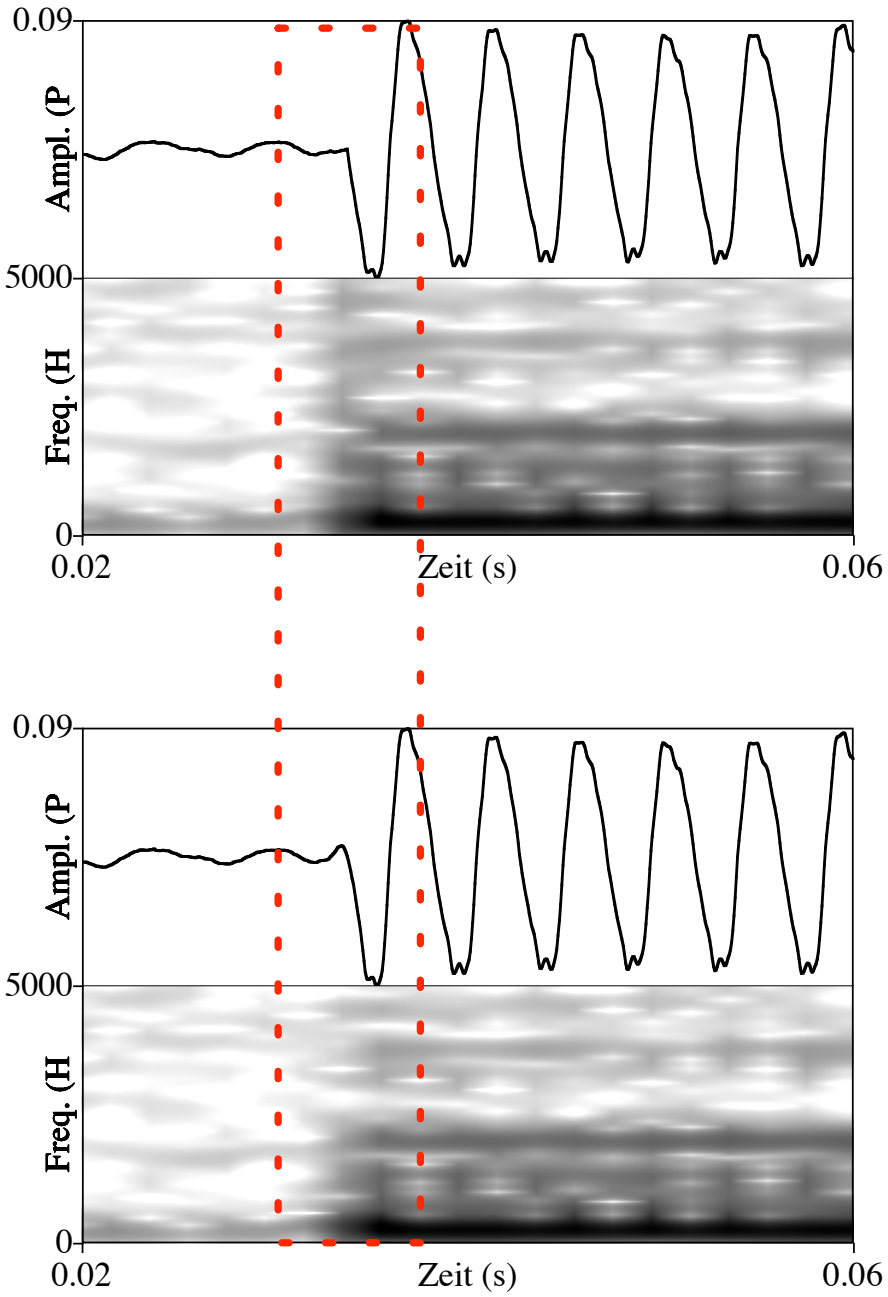

Abbildung 5.2.6.: Verkettung der Phone [ɛ] und [n] ohne (oben) und mit Glättung (unten).

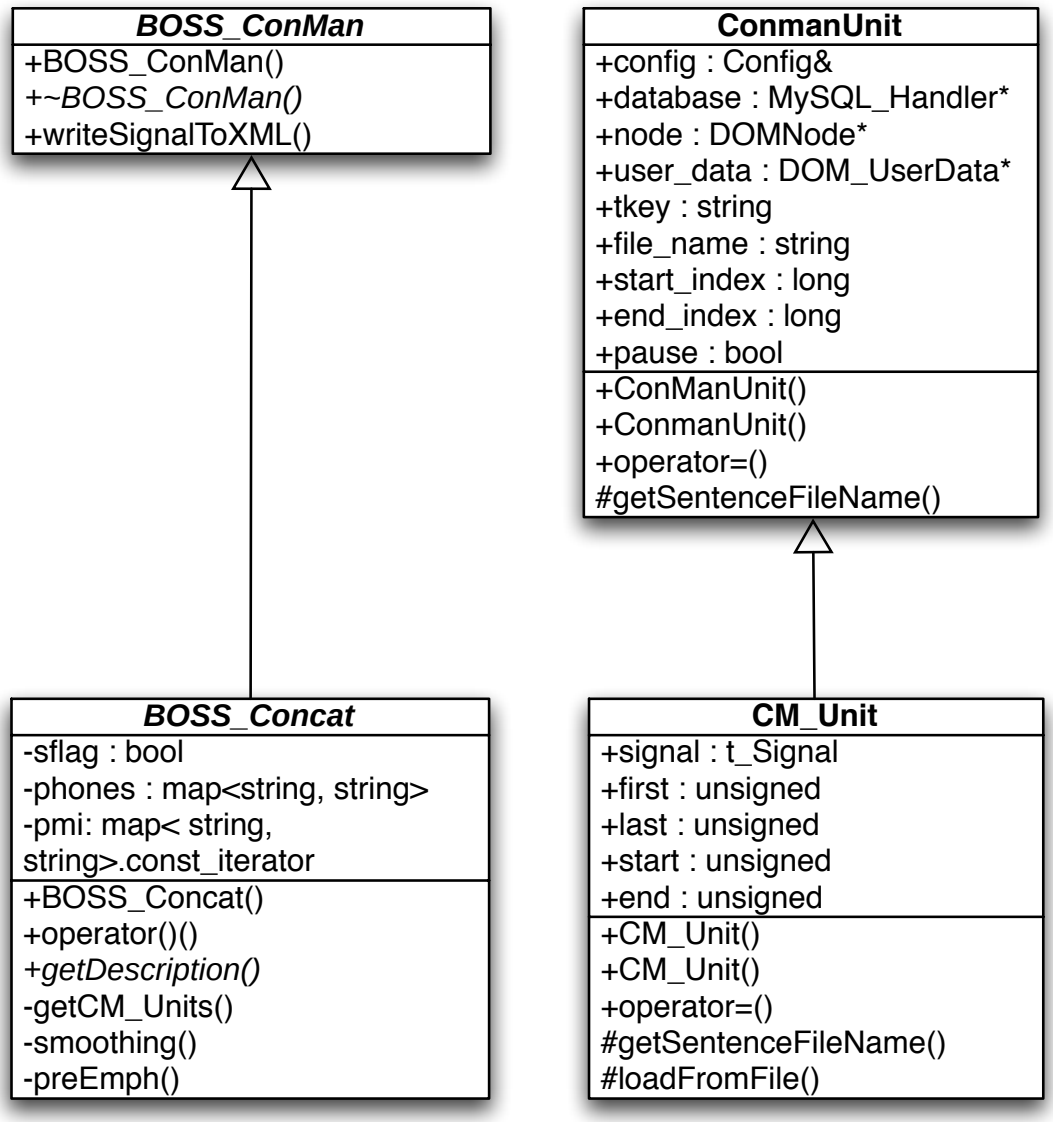

Abbildung 5.2.7.: UML-Diagramm der BOSS-Klassen für die Konkatenation und Manipulation der Sprachsignale.

Rückgabe des Signals notwendig und `BOSS_ConMan` kann, wie alle anderen Module, eine von `BOSS::Module` abgeleitete Klasse sein, die vom Scheduler dynamisch geladen und aufgerufen wird[10]. `ConmanUnit` repräsentiert das Sprachsignal einer Einheit und enthält Methoden und Attribute zur Dateiermittlung aus der Datenbank, sowie Extraktion, Speicherung und Beschreibung des Signals. Beide definieren den kleinsten gemeinsamen Nenner an Funktionalität für die jeweilige Aufgabe. Alle

[10]Dies war auch im `Signalloader` der Fall, war aber zwischenzeitlich wegen Fehlern in der XML-Bibliothek durch eine konventionelle Wertrückgabe ersetzt worden.

neuen Manipulations- und Konkatenationsmodule können auf diesen Klassen aufbauen. Die Abbildung zeigt dies für das Modul `BOSS_Concat` und die Klasse `CM_Unit`, die nunmehr das oben geschilderte Glättungsverfahren implementieren bzw. die dazu benötigten Metadaten zu den Segmenten enthalten. Analog sind `BOSS_OLA` und `OLA_Unit` Ableitungen der Basisklassen. Ein Wechsel zwischen einer Synthese mit prosodischer Manipulation oder einfacher Glättung kann mit der Neuimplementierung daher einfach durch den Austausch eines Bibliotheksnamens in der Modulkonfigurationsdatei erfolgen.

5.3. Unit Selection

Auch in der Implementierung der Einheitenauswahl waren Daten und Algorithmen teilweise noch zu stark verzahnt und konnten daher nur durch Umschreiben aller zugehörigen Klassen an andere Anwendungen und Sprachen angepasst werden. Im Folgenden werden die Änderungen, die zur Flexibilisierung durchgeführt wurden, und eine Reihe von Erweiterungen der Unit Selection um neue Funktionen und Eigenschaften beschrieben.

5.3.1. Einbindung der Kontextklassen

Wie in Abschnitt 3.4 geschildert, wurde durch die Multiphon-Spezifikation *phoxsy* für das Deutsche die Anzahl der möglichen Triphon-Kontexte explosionsartig gesteigert. Daher wurden die Kontextklassen (vgl. Abschnitt 3.2.3) eingeführt. Diese sorgen dafür, dass alle Multiphon-Einheiten, die denselben Anfangslaut besitzen, als äquivalenter rechter Kontext für eine Einheit betrachtet werden können. Gleiches gilt für Endlaute und den linken Kontext. In einer zweiten Stufe wird von der Lautidentität abstrahiert, und es werden nur noch die Artikulationsorte der Anfangs- und Endlaute betrachtet. Die Zuordnung von Phonen und Multiphonen zu ihren linken und rechten Kontextklassen erster und zweiter Stufe wird über eine Tabelle vorgenommen (s. Tabelle A.2.1 im Anhang). Diese Tabelle wird von der eigens geschaffenen Klasse `BOSS_context` in ein Map-Objekt eingelesen und die Abbildungsinformationen in verschiedenen Programmteilen von BOSS zur Verfügung gestellt. So wurde das Tool `addcontext` mit Hilfe dieser Klasse dahingehend erweitert, außer dem expliziten phonetischen Kontext auch die Kontextklassen für jede Einheit einzutragen. Diese Kontextklassen-Attribute werden in Tabelle 5.3.1 beschrieben. Zur Laufzeit der Synthese ist der sprachenunabhängige Teil des Transkriptions-Moduls für die Eintragung des phonetischen Kontextes auf allen Auswahlebenen zuständig. Auch hier wurde die `BOSS_context`-Klasse eingebaut, um gleichzeitig

auch die Kontextklassen-Attribute setzen zu lassen.

Attribut	Bedeutung
`CCLeft`	linke Kontextklasse des vorangehenden (Multi-)Phons (1. Stufe),
`CCRight`	rechte Kontextklasse des folgenden (Multi-)Phons (1. Stufe)
`CCLeft2`	linke Kontextklasse des vorangehenden (Multi-)Phons (2. Stufe)
`CCRight2`	rechte Kontextklasse des folgenden (Multi-)Phons (2. Stufe)

Tabelle 5.3.1.: Attribute zur Speicherung der Klassenzuordnungen des segmentalen Kontextes.

Wie in Abschnitt 4.2.4 geschildert, findet in BOSS eine Vorauswahl vor der Unit Selection statt, um den längstmöglichen Einheitentyp auszuwählen und nur besonders geeignete Kandidaten durch die zeitaufwändige Einheitenauswahl verarbeiten zu lassen. Dabei ist der phonetische Kontext ein wichtiges Kriterium. Durch den Einbau der Kontextklassen wurde es möglich, die Pre-Selection dahingehend zu verändern, dass zwischen der Vorauswahl von Kandidateneinheiten im exakten phonetischen Kontext oder einem völlig beliebigen Lautkontext nun auch die Möglichkeit besteht, Kandidaten mit zumindest ähnlicher Lautumgebung auszuwählen. Anders formuliert muss nun nicht mehr so häufig zwischen einer Treffermenge von 0 für Multiphonkontexte oder einer Menge, die der Anzahl aller *tokens* des Einheitentyps entspricht, gewählt werden.

5.3.2. Konfigurierbare Vorauswahl von Einheiten

In der in Abschnitt 4.2.4 vorgestellten Version der Pre-Selection waren die Auswahlkriterien und die Anzahl und Art der durchgeführten Datenbankanfragen bevor zum nächstkleineren Einheitentyp gewechselt wurde, im Quellcode fest verankert. Für eine multilinguale und multifunktionale Anwendung, wie sie für das BOSS-System angestrebt wurde, war diese Lösung nicht flexibel genug. Daher wurden Überlegungen angestellt, wie der Ablauf der Pre-Selection in Konfigurationsdateien kodiert werden könnte. Das Ergebnis war das in Abbildung 5.3.1 gezeigte Format.

Für jede Ebene, die in der Synthese verwendet werden soll, existiert eine Textdatei, die alle Abfragen auf dieser Ebene definiert. Abfragen sind durch Leerzeilen voneinander abgesetzt. Jede Abfrage enthält mindestens eine Eigenschaft einer Einheit, die als Suchkriterium verwendet werden soll. Eigenschaften sind durch je eine Zeile mit zwei Spalten beschrieben. Links ist der Name der Datenbank-Spalte angegeben, die die gesuchte Eigenschaft enthält, rechts das Attribut, das die Eigenschaft im

XML/DOM beschreibt. Im Idealfall, wie auch im Beispiel, sind die Spalten- und Attributnamen für eine Eigenschaft identisch.

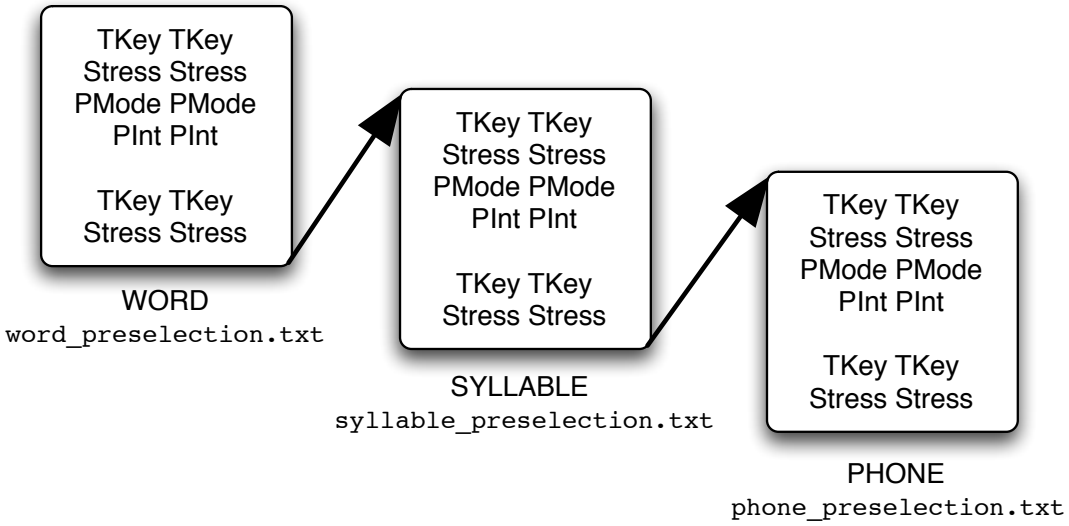

Abbildung 5.3.1.: Einfaches Beispiel für die Konfiguration der Einheitenvorauswahl.

Die in der Abbildung gezeigte vereinfachte Konfiguration würde demnach zunächst auf Wortebene nach solchen Einheiten in der Datenbank suchen lassen, deren Wert in der Spalte `TKey` dem Wert im Attribut `TKey` der prädizierten Äußerungsstruktur entspricht. Gesucht würden also solche Einheiten, deren kanonische Transkription mit der Transkription der Zieläußerung übereinstimmt. Weitere Kriterien für die Auswahl wären hier der Grad der lexikalischen Akzentuierung (`Stress`) und die Phrasenposition und -art (`PMode, PInt`). Werden keine derartigen Wort-Einheiten gefunden, wird nach Wörtern gesucht, die zumindest der kanonischen Transkription und der Akzentuierungsstufe der Vorgabe entsprechen. Bleibt auch dies ohne Ergebnis, wird die Suche auf der nächsten Ebene fortgesetzt. Da im Beispiel die Konfigurationen für alle Ebenen identisch sind, wiederholt sich derselbe Ablauf, bis passende Einheiten gefunden werden.

Die Flexibilisierung der Pre-Selection war die Grundlage für eine umfassende Erweiterung der Vorauswahl-Hierarchie, die in einem späteren Abschnitt (6.3.3) diskutiert wird.

Ein zusätzlicher Nutzen der beschriebenen Änderungen ergibt sich daraus, dass im Verbund mit der notwendigen Umstrukturierung der Unit Selection, nun Einheitenebenen auch vollständig übersprungen werden können, indem keine oder leere Konfigurationsdateien für diese Ebenen angegeben werden. Dadurch lässt sich mit dem aktuellen BOSS z. B. eine reine Wortsynthese oder, mit den in Abschnitt 5.3.4

beschriebenen Änderungen, auch Halbphonsynthese realisieren.

5.3.3. Kostenfunktionen

Wie in Abschnitt 4.2.4 beschrieben, enthielt der Protoyp der BOSS-Kostenfunktion nur einen einzigen Kostenterm, nämlich die Transitionskostenberechnung des Spektralabstandes an den Konkatenationsstellen. Die Anpassung der Synthese an die Aufgaben in den Projekten PiAVIDA und klickTel führte zur Entwicklung weiterer *subcosts*. Neben dem von Bröggelwirth eingeführten Einheitenkostenterm in Formel 5.3.1 zur Bewertung von Daueruntersch ieden werden hier die eigenen Funktionen zur Berechnung von Einheitenkosten vorgestellt.

$$c = |dur(a) - dur(u)| \tag{5.3.1}$$

Vor der Einführung der konfigurierbaren Pre-Selection war der lexikalische Akzent kein Kriterium in der Vorauswahl von Einheiten. Um diese Eigenschaft trotzdem in der Einheitenauswahl verwenden zu können, wurde die in Formel 5.3.2 dargestellte Kostenfunktion zur Bewertung der Abweichungen zwischen dem prädizierten Wert a (Attribut) und einer Kandidateneinheit u (*unit*) implementiert.

$$c = |f(a) - f(u)| \tag{5.3.2}$$

Da die Werte des zugrundeliegenden `Stress`-Attributes phonetisch und nicht quantitativ definiert sind, d. h. Primärakzent `1` unterscheidet sich stärker vom unakzentuierten Fall `0` als der Sekundärakzent `2`, wird zuvor folgende Abbildung vorgenommen:

$$f(x) = \begin{cases} 1, & \text{wenn } x = 2 \\ 2, & \text{wenn } x = 1 \\ x, & \text{sonst} \end{cases} \tag{5.3.3}$$

Die Phrasierung wurde in BOSS II nur in der Vorauswahl von ganzen Wörtern berücksichtigt, da sie zunächst auch nur für diese definiert war. Nach den Änderungen, die in Abschnitt 5.4.1 beschrieben werden, war es jedoch möglich, auch Kandidaten anderer Einheitentypen nach ihren `PInt`- und `PMode`-Eigenschaften auszusuchen. Dies wurde zunächst ebenfalls durch einen Kostenterm der Unit Selection realisiert:

$$c = \begin{cases} 2, & \text{wenn a } = \texttt{" "} \wedge \text{u } \neq \texttt{" "} \\ 1, & \text{wenn a } = \texttt{"?"} \wedge \text{u } \neq \texttt{"?"} \\ 1, & \text{wenn a } = \texttt{"."} \wedge \text{u } \neq \texttt{"."} \\ 0, & \textit{sonst} \end{cases} \tag{5.3.4}$$

Damit wird die Verwendung phrasenfinaler Einheiten in phrasenmedialer Position als gravierender bewertet als der umgekehrte Fall.

Eine Gewichtung und Aufsummierung von *subcosts* war im BOSS-Prototyp aufgrund fehlender Kostenfunktionen nicht notwendig. Mit der Implementierung der neuen Kostenterme änderte sich dies, und die in Formel 5.3.5 gezeigte Funktion wurde implementiert. Sie entspricht der in Verbmobil verwendeten Aufsummierungsfunktion (vgl. Abschnitt 2.3.1). Die Variable c_i steht hier sowohl für Einheiten- als auch Transitionskosten.

$$c_s = \frac{\sum_{i=1}^{n} w_i c_i}{n} \tag{5.3.5}$$

Eine weitere Kostenfunktion, die speziell für manuell annotierte Inventare erstellt wurde, wird bei den Arbeiten zur Telefonauskunftssynthese in Abschnitt 6.3.4 vorgestellt.

Die Gewichte für die einzelnen *subcosts* lassen sich flexibel über das Konfigurations-Interface einstellen. Als Konvention wurde festgelegt, dass alle Optionen, die Gewichte für Einheitenkosten festlegen, das Präfix *ucw_* (*unit cost weight*) erhalten und denjenigen für Transitionskosten ein *tcw_* (*transition cost weight*) vorangestellt wird. Indem man einen Gewichtungsfaktor auf 0 setzt, kann man verhindern, dass die entsprechende Kostenfunktion in die Gesamtkostenberechnung einfließt. Dies ist so implementiert, dass die entsprechende Funktion nicht etwa 0 zurück liefert, sondern gar nicht erst ausgeführt wird und damit auch nicht in die Zahl der verwendeten Kostenterme n eingeht.

Eine Normierung der Kosten findet derzeit nicht statt, was die manuelle Einstellung von Gewichten erheblich erschwert. Zukünftige Versionen von BOSS sollten daher mit einem Tool ausgestattet werden, das die Wertebereiche der einzelnen Attribute erfasst und Vorgaben für eine Abbildung der Kosten auf einen Wertebereich von bspw. 0 bis 1 erlaubt (vgl. Abschnitt 2.2). Auch fehlt ein Mechanismus zur automatischen Einstellung der Gewichte. Eine halbwegs akzeptable Lösung dieses Problems zu finden böte jedoch vermutlich Stoff für eine weitere Dissertation (s. hierzu auch Abschnitt 2.5.2).

Die für die Kostenberechnung relevanten Klassen in BOSS waren von Anfang an

so modular aufgebaut, dass neue Funktionen ohne Änderungen an der für die Einheitenauswahl zentralen Klasse `BOSS_Unitselection` eingebaut werden konnten. Es sind dies die Klasse `BOSS_Node`, die einen Kandidaten für die Unit Selection im Einheitengraphen repräsentiert, und die Klasse `Cost`[11], die die Einheiten- und Transitionskostenfunktionen enthält. Die Datenstrukturen und Algorithmen dieser Klassen können als sprachen-, korpus- und anwendungsspezifisch betrachtet werden. Daher war es nicht ausreichend, die Klassen nur ohne Umschreiben der Unit Selection verändern zu können. Es fehlte auch die Möglichkeit, verschiedene Implementierungen dieser Klassen ohne Neukompilierung von `BOSS_Unitselection` ausführen zu können. Daher wurde der Unit-Selection-Code von Holčík mit großem Aufwand so umgeschrieben, dass nun, ähnlich wie im Module Scheduler `BOSS_Synthesis`, eine durch den Sprachkode des verwendeten Korpus vorgegebene spezifische Version von *BOSS_Node* und *Cost* geladen wird.

5.3.4. Verarbeitung von Halbphon-Einheiten

Stöber schreibt in seiner Dissertation, dass mit BOSS auch die ,,klassische Diphonsynthese" (2002, S. 163) realisiert werden kann. Einschränkend nennt er allerdings die bis dahin noch nicht erfolgte Erweiterung der Auswahlebenen um Halbphone als Grundvoraussetzung für die Integration der Diphonsynthese (S. 178). Dies ist unmittelbar einsichtig, betrachtet man die strenge Hierarchie der Ebenen, die auch durch die Wahl der XML-Repräsentation eine Definition von Einheiten über die Grenzen der übergeordneten Ebenen nicht ohne Weiteres erlaubt. Der Weg zum Diphon muss also zunächst über eine Aufteilung der Phone innerhalb der Grenzen des Segmentes führen.

Die Erweiterung um Halbphone bezeichnet Stöber als ,,aufgrund des modularen Aufbaus des *Unit Selection*-Moduls unproblematisch" (2002, S. 178). In der Praxis stellte sich die Integration der neuen Auswahlebene dann auch als nicht schwierig, aber durchaus aufwändig heraus, da nicht nur die Unit Selection, sondern auch die mittlerweile gestiegene Zahl an Synthesemodulen und Korpuswerkzeugen und die Annotationsdatenbank z. T. größere Anpassungen erforderten. Weniger trivial waren hingegen die Entscheidungen, wie Halbphone in der Einheitenauswahl zu behandeln sind und wie die Beschreibung dieser Einheiten sich zu den Attributen der übergeordneten Phone verhält. Im Folgenden wird die Implementierung der Halbphon-Ebene in BOSS beschrieben.

[11] vormals etwas unglücklich *UNIT_Cost* genannt, aber seit jeher sowohl für Einheiten- als auch Transitionskostenberechnung zuständig.

5.3.4.1. Einbettung in die XML-Repräsentation

Die Vokabulare für BOSS-Inventar- und BOSS-Server-XML wurden jeweils dergestalt um eine Elementebene `<HALFPHONE>` erweitert, dass jedes `<PHONE>`-Element entweder keine oder genau zwei Halbphone enthalten darf. Die Attribute eines `<HALFPHONE>`-Elements entsprechen denen des zugehörigen Phons. Tabelle 5.3.2 zeigt die Beziehung zwischen den Werten dieser Attribute auf der Phon- und Halbphonebene.

<PHONE>		<HALFPHONE>		
Inventar-Attribut	Server-Attribut	Attribut	Beziehung	
			1. Hälfte	2. Hälfte
TKey	TKey	TKey	identisch	identisch
TReal	—	TReal	identisch	identisch
First	—	First	identisch	Last $-\frac{\text{Last}-\text{First}}{2}$
Last	—	Last	First $+\frac{\text{Last}-\text{First}}{2}$	identisch
—	Dur	Dur	$\frac{\text{Dur}}{2}$	$\frac{\text{Dur}}{2}$
Stress	Stress	Stress	identisch	identisch
PMode	PMode	PMode	identisch	identisch
PInt	PInt	PInt	identisch	identisch
CLeft	CLeft	CLeft	identisch	TKey 1. Hälfte
CRight	CRight	CRight	TKey 2. Hälfte	identisch
—		Half	1	2

Tabelle 5.3.2.: Beziehung zwischen den Attributen und Attributwerten der `<PHONE>`- und `<HALFPHONE>`-Ebenen in den Repräsentationen für Server und Inventar.

Die in Abschnitt 5.3.1 vorgestellten Kontextklassenattribute verhalten sich analog zu `CLeft` und `CRight` in der Tabelle. Die vollständigen Schemata der BOSS-Inventar- und Server-XML-Dokumente finden sich im Anhang in den Listings C.1 und C.2.

5.3.4.2. Erweiterung der Korpusaufbereitung und der Datenbank

Um die Halbphonebene mit möglichst geringem Aufwand in die Korpusannotation einzubetten, wurde das Programm `addhp` zur Reihe der Inventarwerkzeuge (vgl. Abschnitt 4.1) hinzugefügt. Es kopiert alle Attribute der Phon-Ebene auf die von ihm angelegte Halbphon-Ebene und passt die Kontext- und Kontextklassenwerte gemäß Tabelle 5.3.2 an. Als Endwert des ersten Halbphons und Startwert des zweiten wird die Mitte des Phons festgelegt. Um in der Vorauswahl der Unit Selection zwischen der

ersten und zweiten Hälfte eines Phons unterscheiden zu können, wird außerdem das Attribut `Half` auf den Wert `1` oder `2` gesetzt. Alternativ zur automatischen Setzung des Schnittpunktes kann `addhp` auch Grenzmarker aus speziellen *BOSS halfphone files* (BHP) einlesen und damit auch manuell oder durch ein anderes Programm erzeugte asymmetrische ,,Halb"phone verarbeiten.

Das Format einer BHP-Datei entspricht der ersten Spalte eines BLF und besteht nur aus Samplewerten zur Markierung der Schnittpunkte innerhalb der Phone. Die Grenzen der Phone werden nicht annotiert. Die zweite Spalte kann optional ein beliebiges Symbol oder eine Zeichenfolge enthalten. Dies ist z.B. für die manuelle Annotation mit *wavesurfer* wichtig, das eine Beschriftung von Labelgrenzen zwingend vorschreibt. Damit wird das BHP-Format kompatibel mit den in `wavesurfer` implementierten Ein- und Ausgaberoutinen für BLF. In der Verarbeitung durch `addhp` wird die zweite Spalte ignoriert.

Da die Halbphone die Attribute der übergeordneten Phonebene erben, ist es wichtig, dass `addhp` als letztes Tool der Informationen ergänzenden Programme aufgerufen wird, aber vor den signalbezogenen Schritten zur Optimierung der Einheitengrenzen (`optbounds`) und der Mel-Cepstrum-Berechnung (`melbounds`). In der Darstellung der Abfolge in Abbildung 4.1.2 wäre dies die Position zwischen Schritt 2 und 3. Die Programme `optbounds` und `melbounds` arbeiten die XML-Hierarchie der Auswahlebenen ohne Kenntnis der Benennung dieser Ebenen ab und führen auf allen Einheiten die gleichen Funktionen aus. Daher war an diesen Tools keine Anpassung erforderlich. Gleiches gilt für das später hinzugefügte `addf0` (s. Abschnitt 4.2.4). Ähnlich generisch ist das Programm `blfxml2db` für den Import in die relationale Datenbank aufgebaut. Die Anpassung beschränkte sich hier auf das Hinzufügen des Strings `HALFPHONE` zur Liste der verwendeten Auswahlebenen-Namen.

Der Datenbank-Struktur mussten für die Halbphon-Erweiterung lediglich zwei Tabellen hinzugefügt werden: `halfphone_data`, die bis auf die zusätzliche *Half*-Spalte vom Aufbau identisch mit *phone_data* ist, und `phone_map`, die die Beziehung zwischen der Phon- und der Halbphonebene herstellt (vgl. Abschnitt 4.1.3). Das vollständige SQL-Schema der Datenbank ist in Listing D.1 im Anhang zu finden.

5.3.4.3. Veränderungen an den Laufzeitkomponenten

Da im BOSS-Server die phonetische Struktur einer Zieläußerung durch das Transkriptionsmodul festgelegt wird, wurde entschieden, auch die Halbphon-Elemente an dieser Stelle einzufügen. Der Einbau war durch Hinzufügen einer einzigen Methode im sprachenunabhängigen Teil des Moduls zu bewerkstelligen. Diese geht bei der Verarbeitung der Phone und Halbphone, soweit möglich, analog zu `addhp` vor.

Welches der beste Weg für die Anpassung des Dauerprädiktionsmoduls wäre, war dagegen weniger eindeutig festzustellen. Da die Vorhersage auf Phon-Basis stattfindet, existiert ein Schätzwert für die Länge eines Halbphons nur dann in gleicher Qualität wie die Lautdauer-Prädiktion, wenn im Korpus symmetrisch aufgeteilte Halbphone verwendet werden. Ist dies nicht der Fall, wird durch die Annahme $dur(hp) = \frac{dur(p)}{2}$ ggf. ein größerer Fehler eingefügt. Trotzdem wurde dieser Weg gewählt, weil eine Umstellung der Dauerprädiktion auf Halbphone aus Gründen der Abwärtskompatibilität zu rein phonbasierten Inventaren nicht sinnvoll erschien. Für die Zukunft wäre zu überlegen, ob das Dauermodul um eine zusätzliche Halbphonprädiktion erweitert werden sollte oder durch Heuristiken von typbezogenen relativen Schnittpunktpositionen für ein bestimmtes Korpus eine exaktere Aufteilung des Phondauerwertes ermöglicht werden könnte. Aus Mangel an asymmetrisch annotierten Halbphonen konnte bisher allerdings nicht getestet werden, ob der entstehende Fehler wirklich groß genug ist, um solche Maßnahmen zu rechtfertigen.

Durch die Vorarbeiten zur Flexibilisierung der Vorauswahl (Abschnitt 6.3.3) beschränkten sich die Änderungen am Unit-Selection-Modul im Großen und Ganzen auf die Definition einer Konfigurationsdatei für Halbphone und das Laden derselben. Die Kostenfunktionen mussten nicht angepasst werden.

Das Konkatenations- und Manipulationsmodul entstand zeitgleich mit dem Einbau der Halbphone und war daher von Anfang an für diese Auswahlebene ausgelegt, bzw. ausreichend generisch konzipiert. Die Signalloader-Funktionen in `BOSS_ConmanUnit` enthalten daher im Vergleich zum obsoleten `BOSS_Signalloader` ein zusätzliches SQL-Statement zur Ermittlung der Ursprungsdatei einer Einheit (vgl. 4.3):

```
1 -- Halbphon-Einheiten
2 SELECT sentence_data.File FROM sentence_data, sentence_map, word_map,
      syllable_map
3 WHERE phone_map.HalfphoneNum = "halfphone_num"
4 AND   sentence_data.Num = sentence_map.SentenceNum
5 AND   word_map.WordNum = sentence_map.WordNum
6 AND   syllable_map.SyllableNum = word_map.SyllableNum
7 AND   phone_map.PhoneNum = syllable_map.PhoneNum;
```

Listing 5.1: SQL-Query zur Ermittlung der Herkunftsdatei einer Halbphoneinheit.

5.3.4.4. Überlegungen zur Erweiterung auf Diphon-Synthese

Stöber (2002) lässt offen, wie BOSS auf Basis der Halbphone zu einer Diphon-Synthese ausgebaut werden soll. Hierfür gibt es allerdings eine Vielzahl von Möglichkeiten: Für eine klassische Diphonauswahl müsste, wollte man das Unit-Selection-

Modul nicht komplett umschreiben oder austauschen, die Vorauswahl zunächst auf Halbphone eingeschränkt und dort ausschließlich diejenigen Auswahlkriterien verwendet werden, für die bekanntermaßen alle möglichen Varianten als Diphone im Korpus existieren. Nur so kann garantiert werden, dass in einer Menge von Kandidaten für ein Endhalbphon[12] A und in einer Menge von Kandidaten für ein darauf folgendes Anfangshalbphon B auch zwei Einheiten enthalten sind, die im Inventar ein Diphon bilden. Damit wird sich die Kriterienwahl i. A. auf die Lautidentität und den rechten bzw. linken Kontext beschränken, da die Anzahl der benötigten Halbphonfolgen quadratisch mit der Zahl der möglichen Parameterwerte wächst. Bereits das Hinzufügen des Parameters `Stress` mit seinen drei möglichen Merkmalsausprägungen würde für jede segmental unterschiedliche Folge neun unterschiedliche Varianten erfordern, die garantiert im Korpus vorhanden sein müssten. Abbildung 5.3.2 illustriert einen Fall, in dem die gewünschte Sequenz von lexikalischen Akzenten zwar in Form von Einzelhalbphonen realisierbar wäre, aber nicht als Diphon im Inventar vorhanden ist.

Weiterhin müsste in der Unit Selection sichergestellt werden, dass die Berechnung des optimalen Pfades durch den Auswahlgraphen an potenziellen Diphongrenzen immer die Verbindung zwischen zwei Diphonbestandteilen wählt. Dies kann entweder erreicht werden, indem der Optimierungsalgorithmus dementsprechend modifiziert wird oder in der Berechnung der Transitionskosten Sorge getragen wird, dass die aufsummierten Einheiten- und Transitionskosten an den Kanten zwischen zwei einem Diphon angehörigen Halbphonen immer unter denen der anderen Kanten liegt.

Eine andere Möglichkeit wäre, eine aufgeweichte Form der Diphonsynthese innerhalb des Multilevel-Ansatzes von BOSS zu verfolgen. Beispielsweise könnte sich die Auswahl auf die Satz-, Silben- und Halbphonebenen beschränken. Die Vorauswahl würde dabei unverändert bleiben und nur die Phon-Ebene ausklammern. Die Unit Selection selber würde so gestaltet wie oben geschildert, so dass Diphon-Übergänge zwar nur zufällig in die Unit Selection gelangen, dann aber in der Berechnung der Ausgabefolge bevorzugt würden. Dies ähnelt dem Konsekutivitätskriterium in anderen Unit-Selection-Systemen, wäre aber auf die potenziellen Diphon-Übergänge beschränkt. An diesen Stellen ist die Konsekutivität besonders wichtig, weil sich an ihnen die für die Perzeption kritischen Lauttransitionen abspielen.

Momentan erlaubt BOSS die Synthese von Diphoneinheiten noch nicht. Auch fehlen formale Tests zur Qualität der Halbphonsynthese. Ein am IfK laufendes Dissertationsvorhaben von Schwill will daher klären, ob und wie sich die Qualität durch verschiedene Kombinationen bekannter Syntheseeinheiten wie Halbphon, Diphon und Triphon verändert.

[12] Anfang und Ende beziehen sich hier auf das Phon, nicht auf das Diphon.

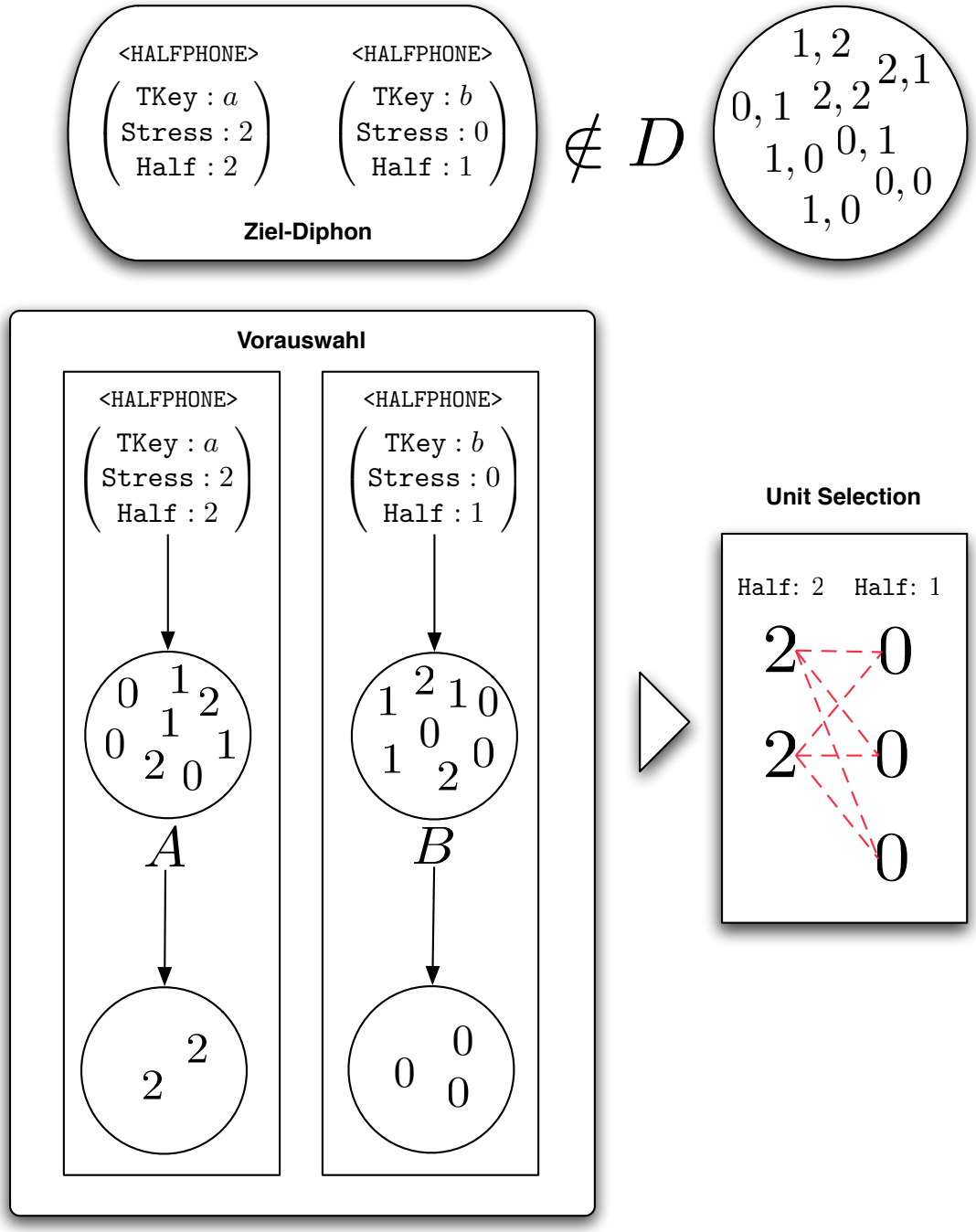

Abbildung 5.3.2.: Beispiel für eine gescheiterte Auswahl von Diphonen nach Halbphon-Vorauswahl. D ist die Menge aller Diphone der Lautfolge a, b. Dargestellt sind nur die jeweiligen Werte des `Stress`-Attributs. Die gesuchte Folge a, b ist mit den gewünschten `Stress`-Parametern als Diphon nicht in der Datenbank vorhanden. Die auf Halbphonen basierende Pre-Selection wählt pro Halbphon eine nach den Vorgaben optimale Einheit aus. Dadurch wird eine Auswahl eines Diphons der Folge unmöglich, was durch die gestrichelten roten Kanten im Unit-Selection-Graphen symbolisiert wird.

5.4. Prosodische Symbolverarbeitung

5.4.1. Behandlung von Phrasengrenzen

BOSS nutzt die Attribute `PMode` und `PInt` (s. Tabelle 4.1.1), um zu markieren, ob sich eine Einheit vor einer Phrasengrenze befindet und welcher Art diese Grenze ist. Anfänglich waren diese Attribute allerdings nur auf der Wortebene gesetzt und die Einheitenauswahl berücksichtigte die Phrasenposition daher nur für die Selektion von Wörtern. Da die prosodischen und segmentalen Auswirkungen von Grenzpositionen, wie phrasenfinale Längung und Änderung der Stimmqualität, aber auf allen Auswahlebenen relevant sind, sollte die Annotation auch auf die Einheiten unterhalb des Wortes ausgedehnt werden. Dabei ergab sich die Frage, ob die Silben und Phone über die ganze Länge eines Wortes perzeptiv betroffen sind oder nur ein Teil davon als Grenzeinheiten markiert werden sollte. Schwill (2001) untersuchte den Einflussbereich der Grenzposition und kam zu dem Schluss, dass im Allgemeinen nur die letzte Silbe eines Grenzwortes akustische Auswirkungen aufweist.

Aufgrund dieser Erkenntnisse wurde das Transkriptionsmodul um eine Funktion erweitert, die die Attribute eines Wortes an der Phrasengrenze auf die letzte Silbe des Wortes und deren Untereinheiten propagiert. Die anderen Silben werden so markiert, als wären sie phrasenmedial gesprochen. Zudem wurde ein Programm *addphrases* zur Korpusaufbereitung geschrieben, das eine analoge Verarbeitung der Inventar-XML-Dateien vornimmt.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <TEXT>
3    <SENTENCE SentenceID="1" TargetFile="nager_080208-165918_080208-165918"
         TargetPath="." Type="?">
4      <WORD CCLeft="$p" CCLeft2="$p" CCRight="$p" CCRight2="$p" CLeft="$p"
           CRight="$p" Dur="743" Orth="Nager" PInt="5" PMode="?" PNr="4" TKey="
           na:g6">
5        <SYLLABLE CCLeft="$p" CCLeft2="$p" CCRight="g" CCRight2="VEL" CLeft="$
             p" CRight="g" Dur="244" PInt="0" PMode="" PNr="2" Stress="1" TKey=
             "na:">
6          <PHONE CCLeft="$p" CCLeft2="$p" CCRight="a" CCRight2="CEN" CLeft="$p
               " CRight="a:" Dur="111" PInt="0" PMode="" Stress="0" TKey="n">
7            <HALFPHONE CCLeft="$p" CCLeft2="$p" CCRight="a" CCRight2="CEN"
                 CLeft="$p" CRight="a:" Dur="56" Half="1" PInt="0" PMode=""
                 Stress="0" TKey="n"/>
8            <HALFPHONE CCLeft="$p" CCLeft2="$p" CCRight="a" CCRight2="CEN"
                 CLeft="$p" CRight="a:" Dur="56" Half="2" PInt="0" PMode=""
                 Stress="0" TKey="n"/>
9          </PHONE>
```

```
10          <PHONE CCLeft="n" CCLeft2="ALV" CCRight="g" CCRight2="VEL" CLeft="n"
                CRight="g" Dur="133" PInt="0" PMode="" Stress="1" TKey="a:">
11            <HALFPHONE CCLeft="n" CCLeft2="ALV" CCRight="g" CCRight2="VEL"
                  CLeft="n" CRight="g" Dur="67" Half="1" PInt="0" PMode="" Stress
                  ="1" TKey="a:"/>
12            <HALFPHONE CCLeft="n" CCLeft2="ALV" CCRight="g" CCRight2="VEL"
                  CLeft="n" CRight="g" Dur="67" Half="2" PInt="0" PMode="" Stress
                  ="1" TKey="a:"/>
13          </PHONE>
14        </SYLLABLE>
15        <SYLLABLE CCLeft="a" CCLeft2="CEN" CCRight="$p" CCRight2="$p" CLeft="
              a:" CRight="$p" Dur="499" PInt="5" PMode="?" PNr="2" Stress="0"
              TKey="g6">
16          <PHONE CCLeft="a" CCLeft2="CEN" CCRight="6" CCRight2="CEN" CLeft="a:
                " CRight="6" Dur="188" PInt="5" PMode="?" Stress="0" TKey="g">
17            <HALFPHONE CCLeft="a" CCLeft2="CEN" CCRight="6" CCRight2="CEN"
                  CLeft="a:" CRight="6" Dur="94" Half="1" PInt="5" PMode="?"
                  Stress="0" TKey="g"/>
18            <HALFPHONE CCLeft="a" CCLeft2="CEN" CCRight="6" CCRight2="CEN"
                  CLeft="a:" CRight="6" Dur="94" Half="2" PInt="5" PMode="?"
                  Stress="0" TKey="g"/>
19          </PHONE>
20          <PHONE CCLeft="g" CCLeft2="VEL" CCRight="$p" CCRight2="$p" CLeft="g"
                CRight="$p" Dur="311" PInt="5" PMode="?" Stress="0" TKey="6">
21            <HALFPHONE CCLeft="g" CCLeft2="VEL" CCRight="$p" CCRight2="$p"
                  CLeft="g" CRight="$p" Dur="155" Half="1" PInt="5" PMode="?"
                  Stress="0" TKey="6"/>
22            <HALFPHONE CCLeft="g" CCLeft2="VEL" CCRight="$p" CCRight2="$p"
                  CLeft="g" CRight="$p" Dur="155" Half="2" PInt="5" PMode="?"
                  Stress="0" TKey="6"/>
23          </PHONE>
24        </SYLLABLE>
25      </WORD>
26    </SENTENCE>
27  </TEXT>
```

Listing 5.2: Server-XML-Dokument nach Einfügen der Phrasenstruktur

Listing 5.2 zeigt die Ausgabe des Transkriptionsmoduls für das Wort *Nager*. Zusätzlich zur segmentalen Beschreibung (`TKey`) wurden die Attribute zur Beschreibung des phonetischen Kontextes (`CCLeft, CCLeft2, CCRight, CCRight2, CLeft, CRight`, vgl. Abschnitte 4.2.3 und 3.2.3) und die Anzahl der Phone in Wort und Silbe (`PNr`) gesetzt. Die auf der Wortebene annotierten Werte von `"?"` für `PMode` und `"5"` für `PInt` markieren eine Position des Wortes vor einer Phrasengrenze mit Frageintonation (Zeile 4). Die Werte für `PInt` und `PMode` der ersten Silbe und ihrer Untereinheiten

entsprechen der Notation für phrasenmediale Einheiten (Zeilen 5-14). In der zweiten und letzten Silbe sind sie jedoch identisch mit den Werten des übergeordneten Wortes (Zeilen 15-24).

5.4.2. Anpassung der Pausenstruktur

In BOSS II gab es zunächst keine Möglichkeit, über die Eingabe Einfluss auf die Länge oder die Platzierung von Sprechpausen zu nehmen. Pausen wurden im `BOSS_Signalloader` automatisch an allen Phrasengrenzen gesetzt. Die Längen waren fest im Programmtext vorgegeben und richteten sich nach der Art der Grenze. Verschiedene Erwägungen führten dazu, dass dieser Automatismus entfernt wurde und weitgehende Umstrukturierungen an vielen Modulen inklusive der Unit Selection und Konkatenation vorgenommen wurden, um eine flexible Setzung unterschiedlich langer Pausen zu ermöglichen.

Motiviert war der Umbau in erster Linie dadurch, dass eine adäquate Pausenstruktur wesentlich zur Verständlichkeit beiträgt. In den eigenen Arbeiten wurde dies bei der lautsprachlichen Generierung der Adressbucheinträge im Rahmen des Telefonauskunftsprojektes deutlich (Kapitel 6). Auch die Natürlichkeit profitiert von einer dynamischen Pausensetzung. Goldsmith (1999) schreibt:

> "While these are functions that a TTS device does not absolutely need to duplicate (though perhaps it should sometimes emphasize certain words and phrases), still the fact remains that human listeners expect pauses when they listen to speech, and a functional TTS system must give its listeners those expected pauses. Without them, the task of listening to extended synthetic speech becomes a burdensome task, and the listener's attention will rebel."

In der überarbeiteten BOSS-Version werden Pausen wie Wort-Einheiten im BOSS-XML behandelt. Zwar werden sie im Gegensatz zu diesen nicht aus dem Inventar ausgewählt, sie lassen sich jedoch ebenso mit einem Dauerwert und Kontextattributen versehen. Im Verkettungsmodul wird im letzten Schritt der Synthese von jeder `CM_Unit`, die eine Pause enthält, eine Stilleperiode der entsprechenden Länge generiert. Ist für die Einheit keine Dauer angegeben, wird, wie in der alten Version, auf feste Standardwerte zurückgegriffen.

Dieser Mechanismus ermöglicht einerseits, Dauern schon im Synthese-Client festzulegen. Dies ist schon in einem reinen TTS-Kontext interessant, weil dadurch sprach- und auch textspezifische Einstellungen berücksichtigt werden können. Wichtiger noch ist die Anwendung in einem CTS-Szenario, in dem für die Pausensetzung wichtige Informationen zur Äußerungsstruktur schon vor der eigentlichen Synthese

bekannt sind (s. Kapitel 6). Andererseits wäre nun auch möglich, die Pausensetzung einem datengetriebenen Prosodiemodul zu überlassen, wie es beispielsweise Hao et al. (2006) präsentieren.

Listing 5.3 zeigt die mögliche Ausgabe eines BOSS-Clients für den Text *A, B* in Form eines minimalen XML-Dokumentes (vgl. Abschnitt 4.2.1). Die Pause, in BOSS universell als `$p` notiert, wird in Zeile 5 als eigenständiges Wort-Element hinter der Phrasengrenze nach *A* eingefügt. Über das *Dur*-Attribut wird ihr eine Dauer von 250 ms zugewiesen.

```
1 <?xml version='1.0' encoding='ISO-8859-1' ?>
2 <TEXT>
3   <SENTENCE Type=".">
4    <WORD Orth="A" PInt="2" PMode="?"></WORD>
5    <WORD Orth="$p" Dur="250"></WORD>
6    <WORD Orth="B" PInt="5" PMode="."></WORD>
7   </SENTENCE>
8 </TEXT>
```

Listing 5.3: Minimales Server-XML-Dokument mit Pausenstruktur. Hypothetische Ausgabe einer Client-Applikation.

In Listing 5.4 ist ein Ausschnitt aus dem maximalen XML-Dokument zu sehen, das nach Ablauf aller Syntheseschritte ausgegeben wird. Der Ausschnitt zeigt nur die Elemente, die der Pause angehören. Wie die anderen Wörter auch ist die Pause durch den sprachenunabhängigen Teil des Transkriptionsmoduls gelaufen und dort mit den Untereinheiten Silbe, Phon und Halbphon sowie den Attributen zur Beschreibung des Kontextes und der Phrasenposition ausgestattet worden. Die weiteren Attribute wurden von der Dauerprädiktion eingefügt, die den für die Generierung der Pause ausschlaggebenden `Dur`-Wert auf der Wortebene (Zeile 1) unangetastet gelassen hat.

```
1 <WORD CCLeft="a" CCLeft2="CEN" CCRight="b" CCRight2="BIL" CLeft="?a:"
     CRight="b" Dur="250" Orth="$p" PNr="1" Phrpos="F" TKey="$p">
2   <SYLLABLE Bodi="0" CCLeft="a" CCLeft2="CEN" CCRight="b" CCRight2="BIL"
      CLeft="?a:" CRight="b" Dur="273" PInt="0" PMode="" PNr="1" Phrpos="F"
      Stress="0" TKey="$p">
3     <PHONE CCLeft="a" CCLeft2="CEN" CCRight="b" CCRight2="BIL" CLeft="?a:"
       CRight="b" CRight2="e:" Dur="273" PInt="0" PMode="" Phrpos="F"
       Stress="0" TKey="$p">
4       <HALFPHONE CCLeft="a" CCLeft2="CEN" CCRight="b" CCRight2="BIL" CLeft="
          ?a:" CRight="b" Dur="137" Half="1" PInt="0" PMode="" Phrpos="F"
          Stress="0" TKey="$p"/>
5       <HALFPHONE CCLeft="a" CCLeft2="CEN" CCRight="b" CCRight2="BIL" CLeft="
          ?a:" CRight="b" Dur="137" Half="2" PInt="0" PMode="" Phrpos="F"
          Stress="0" TKey="$p"/>
```

```
6      </PHONE>
7    </SYLLABLE>
8  </WORD>
```

Listing 5.4: Pauseneinheit nach der Verarbeitung durch den BOSS-Server. Ausschnitt aus einem maximalen Server-XML-Dokument.

Mit dem Einbau der Pausen in die Einheitenstruktur konnte auch eine Inkonsistenz in der Unit Selection behoben werden. Zuvor wurden Einheiten, die durch eine satzinterne Phrasengrenze getrennt waren, von den Transitionskostenfunktionen behandelt, als wären sie unmittelbar benachbart. Nichtsdestotrotz wurde bei der Konkatenation an diesen Stellen eine Pause eingefügt. In der neuen Version wird dies vermieden, da bereits zum Zeitpunkt der Einheitenauswahl ein Pausenelement an der Phrasengrenze existiert, das mit den adäquaten akustischen Werten versehen ist.

5.5. Weitere Entwicklungen und Ausblick

Die in diesem Kapitel beschriebenen Änderungen bieten eine solide Basis für die Nutzung von BOSS in neuen Sprach- und Anwendungskontexten, ohne dass die dezentrale Weiterentwicklung eine Einbahnstraße darstellen muss. Das Grundkonzept von BOSS wurde hin zu echter Multilingualität, wie sie das Bell-System für die Diphonsynthese bot (Sproat, 1998), und Multifunktionalität erweitert und die Synthese wurde mit neuen Funktionen ausgestattet, die in vielen denkbaren Szenarien von Nutzen sein können. Natürlich ist das Projekt BOSS damit nicht fertig oder abgeschlossen. Viele Ideen und Konzepte müssen noch erprobt und implementiert werden, und auch die hier vorgestellten Designprinzipien müssen teilweise noch konsequenter in bestehenden BOSS-Modulen umgesetzt werden. Dazu zählen insbesondere Module, die aufgrund fehlender Dokumentation und zeitlicher Beschränkungen nie den Sprung in die aktuelle Version 3 geschafft haben, wie z. B. das Fujisaki-Intonationsmodul von Bröggelwirth und Groß. Gleichzeitig warten die Ergebnisse zahlreicher Magisterarbeiten darauf, in BOSS integriert und weiter ausgebaut zu werden. Da wären beispielsweise die für die Anpassung an die westafrikanische Tonsprache Ibibio entwickelten Erweiterungen der prosodischen Prädiktion und Einheitenauswahl (Bachmann und Breuer, 2007) und die Implementierung einer Variante des Harmonics-plus-Noise-Modells (Rohde und Breuer, 2005). Einen weiteren Beitrag zur prosodischen Manipulation soll die Arbeit von Schönwandt zur Glättung von Grundfrequenzsprüngen liefern. Besonders vielversprechend für den Ausbau von BOSS zum *minimalen multilingualen System* sind die in Planung be-

findlichen Module zur F0-Prädiktion von Jauk und eine Magisterarbeit von Knoben zur Integration von deutschen ToBI-Annotationen (***To****ne and* ***B****reak* ***I****ndices*, bspw. nach Grice et al., 2005) in BOSS. Aufbauend auf der Magisterarbeit von Arnold und den Ergebnissen von Wagner et al. (2000) und Tamburini und Wagner (2007) zur Prominenz und deren automatischer Prädiktion und Detektion könnte dann ein umfassender Vergleich der Performanz akustischer, phonetisch-phonologischer und perzeptiv motivierter Prosodie-Modelle in der Synthese durchgeführt werden. Denn gerade die explizite Modellierung der Prosodie stellt noch einen der Schwachpunkte von BOSS dar. Mit den genannten Erweiterungen könnte dann auch endlich die durch `BOSS_OLA` zwar mögliche, aber mangels Daten nicht verwendete Manipulation der Grundfrequenz genutzt werden. Dies wäre wiederum die Grundlage für weitere Arbeiten zu Schädlichkeit oder Nutzen prosodischer Manipulation in der Unit-Selection-Synthese.

Das beste Beispiel für die Machbarkeit der verteilten Entwicklung an BOSS ist derzeit, die von der Alexander-von-Humboldt-Stiftung zwischen 2004 und 2007 geförderte Institutspartnerschaft zwischen der Phonetik der Adam-Mickiewicz-Universität in Poznań und dem IfK. Viele der Flexibilisierungsmaßnahmen sind in diesem Rahmen durchgeführt worden und zeigen nun ihre positive Wirkung.

Ein Punkt, der zusammen mit den hier geschilderten Designprinzipien dafür Sorge getragen hat, dass die Zusammenarbeit an BOSS leichter geworden ist, wurde bislang nicht erwähnt: die Umstellung von handgeschriebenen Makefiles auf die automatische Generierung von Kompilationsskripten durch die Autobuild-Tools (Vaughan et al., 2000). Diese von Groß, Holčík, Arnold und den Autor durchgeführte Maßnahme, die die Kompilierung des Quellcodes nicht nur auf verschiedenen Linux-Distributionen, sondern auch auf unterschiedlichen Betriebssystemen erheblich erleichterte, war die notwendige Voraussetzung dafür, dass der Quellcode von BOSS Ende 2005, ausgestattet mit der verdienten *major version number* 3, endlich auf dem Open-Source-Portal *Sourceforge* (BOSS, 2005) der Öffentlichkeit zugänglich gemacht werden konnte. In der neuesten Revision kompiliert und läuft BOSS auch auf Windows-Systemen und Mac OS X.

Die Abbildung 5.5.1 bietet im Verbund mit der Darstellung von BOSS II in Abbildung 4.2.1 in Kapitel 4 einen groben Überblick über die Veränderungen und Erweiterungen, die die Version 3 ausmachen.

Einige Gedanken zur Weiterentwicklung des Basis-Systems, wie die Integration von Diphonen, wurden in diesem Kapitel bereits angesprochen. Andere Pläne betreffen insbesondere die weitere Verbesserung der Portabilität und Multilingualität. Z. B. kann BOSS zwar durch die Verwendung von XML-Datenstrukturen Unicode-Zeichen durch das System transportieren, die Stringverarbeitung selbst basiert je-

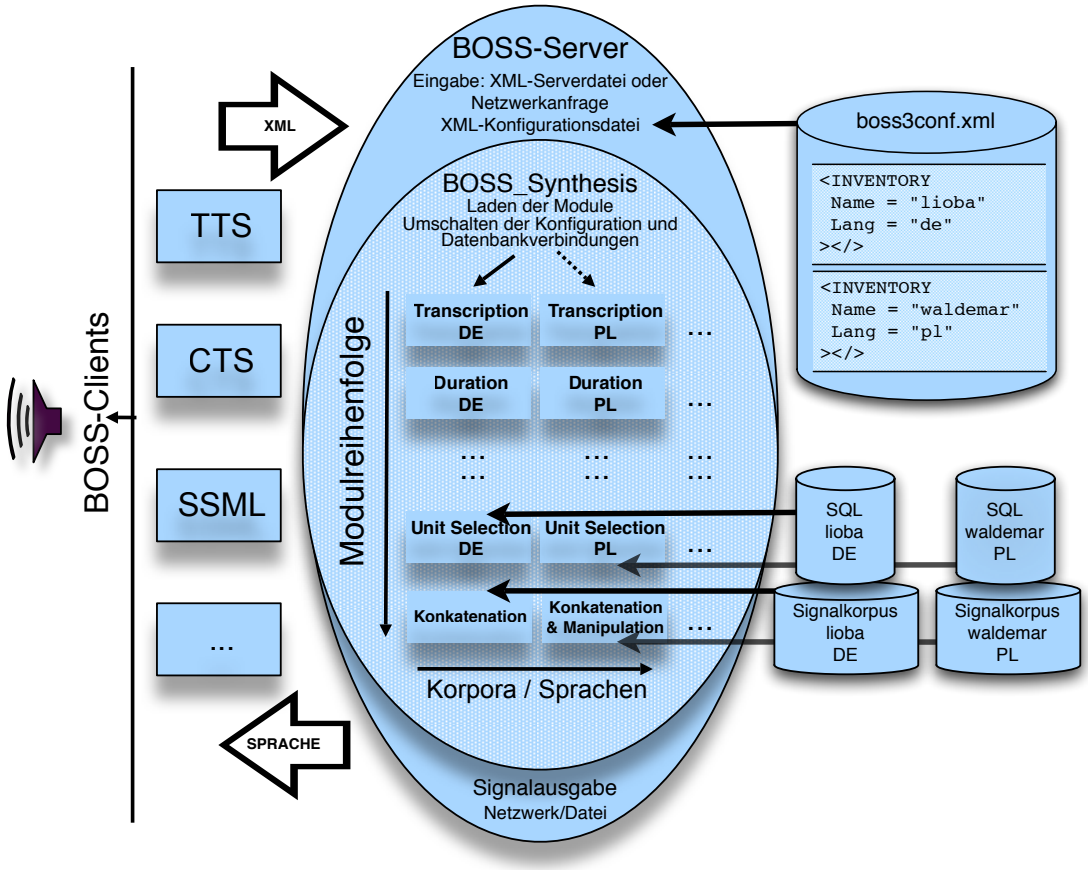

Abbildung 5.5.1.: Architektur des Laufzeit-Systems von BOSS 3.

doch auf den Standard-Klassen von C++ und kommt daher nur mit 8-Bit-kodierten Zeichen zurecht. Auch die Beschränkung auf MySQL als Datenbank ist problematisch, weil die Distribution und Installation für reine Nutzer des BOSS-Systems zu aufwändig ist. Zu guter Letzt bietet der verwendete XML-Parser zwar eine solide DOM-Implementierung, fiel in der Vergangenheit aber öfter durch radikale Umstellungen des API auf, die ebenso einschneidende und zeitaufwändige Veränderungen in BOSS erzwangen. Eine Lösung dieser Probleme könnte sich durch die Verwendung der ebenfalls GPL-lizenzierten Qt-Portabilitätsbibliothek (Trolltech, 2008) ergeben, die Klassen zur Datenbankanbindung an eine Vielzahl von Systemen bietet und Unicode- sowie XML-Funktionen bereitstellt. Damit wäre die Abhängigkeit von nur noch einem Produkt gegeben, was den Aufwand zur Pflege des Codes deutlich reduzieren könnte.

Die schrittweise Erweiterung von BOSS zu einem vollständigen TTS-System hat zur Folge, dass die Architektur nunmehr auch für die Anwendung mit anderen Syntheseverfahren interessant wird. Schließlich sind die grundlegenden Symbolverarbeitungs-

schritte auch für parametrische Verfahren Voraussetzung. Es müssen im Idealfall also nur die Einheitenauswahl und die akustische Synthese durch andere Module ausgetauscht werden. Für die artikulatorische Synthese wurden bereits Versuche durchgeführt, die diesen Aspekt der Multifunktionalität von BOSS demonstrieren (Birkholz et al., 2007). Auch die Einbindung für das aufstrebende Verfahren der HMM-basierten Synthese (Zen et al., 2007) wäre so möglich.

Im folgenden Kapitel wird der Ausbau von BOSS zu einem Adressausgabesystem für die automatische Telefonauskunft beschrieben, der im Rahmen eines Projektes mit der Firma klickTel durchgeführt wurde. Einige der hier aufgeführten Erweiterungen zur Multifunktionalität von BOSS sind Produkte dieser Anpassung. Nachfolgend werden daher nur noch die Komponenten beschrieben, die für die Anwendung in der Auskunft spezifisch waren.

6. Anpassung an eine Telefonauskunftsanwendung

,,Wer zählt die Völker, nennt die Namen, die gastlich hier zusammenkamen?"
Johann Christoph Friedrich von Schiller

,,A name pronounced is the recognition of the individual to whom it belongs. He who can pronounce my name aright, he can call me, and is entitled to my love and service." *Henry David Thoreau*

Der Inhalt eines Telefonbuchs stellt ein Paradebeispiel für das in Abschnitt 2.5.3 geschilderte Problem der großen Anzahl seltener Ereignisse (LNRE) dar und ist damit als Zieldomäne der Sprachsynthese denkbar ungeeignet. Zwar sind in einem Anwendungsszenario, das in der sprachlichen Ausgabe von Adresseinträgen mündet, die syntaktischen Strukturen in den meisten Fällen identisch, dafür stellt die lexikalische Variabilität ungleich höhere Ansprüche an das Korpus und die Symbolverarbeitung, denn bis auf eine relativ kleine Anzahl sehr häufiger Orts-, Straßen-, Familien-, Vor- und Firmennamen stellt jeder Name ein *rare event* dar. Die Aufgabe der Synthese von Namen und Adressen kann also als *phrase-slot-filling* mit praktisch unbeschränkten Füllvarianten angesehen werden. Eines der größten Anwendungsfelder der maschinellen Sprachgenerierung nach diesem Muster stellt die Branche der telefonischen Auskunftsdienste. Unter den Telediensten ist die Telefonauskunft *das* klassische Angebot neben der Zeitansage, so dass es nahe lag, auch für diese Anwendung den Einsatz von Sprachsynthese ins Auge zu fassen.

In den Jahren 2000 bis 2003 wurde am IKP im Auftrag des Unternehmens klickTel GmbH eine auf der BOSS-Synthese aufbauende Telefonauskunftsanwendung entwickelt (Abresch und Breuer, 2004; Breuer und Abresch, 2003), die auch die Namen und Adressen von Teilnehmern ansagen kann. In den folgenden Abschnitten sollen die Schwierigkeiten dieser Aufgabe, die Designentscheidungen für das Korpus und die eingesetzten Algorithmen zur Lösung des Problems näher beschrieben und disku-

tiert werden. Begonnen wird dabei mit den Konzepten für die Struktur des Korpus und der praktischen Verfahrensweise bei der Aufnahme.

6.1. Das Sprachkorpus

In Abschnitt 2.5.3 wurde eine Methode beschrieben, die Textgrundlage von Synthesekorpora über die Lösung eines Mengenüberdeckungsproblems zu kreieren. Die Voraussetzung dafür ist allerdings, dass die Verteilung phonetischer Phänomene der Zieldomäne in großen Teilen bekannt ist. Genau dies war aber zu Beginn des Projektes nicht der Fall, da noch kein Aussprachelexikon für die nur in orthographischer Form vorliegenden Telefonbucheinträge existierte. Somit konnte auch kein automatisches Verfahren zur Transkription eingesetzt werden, das zuverlässig die phonetische Struktur der lexikalischen Bestandteile ermitteln würde. Die Projektplanung sah aber vor, dass schon nach kurzer Zeit ein erstes Testsystem zur Verfügung stehen sollte, dessen Korpus dann sukzessive erweitert werden könnte. Daher wurden zunächst die Lesetexte aus der Dissertation von Portele (1996) von der Sprecherin realisiert und aufgezeichnet, um eine breite Basis von Lauten und Lautfolgen des Deutschen zur Verfügung zu haben. Dann wurden unter Verwendung der häufigsten Namen Lesevorlagen manuell so erstellt, dass die Eigennamen immer an Phrasenpositionen vorkamen, die sie auch in der Synthese der Adresseinträge einnehmen würden. Diese Vorgehensweise bedingte durch das ungünstige Verhältnis zwischen möglicherweise redundanten Inhalts- und Funktionswörtern einerseits und direkt verwendbaren Namenseinheiten, dass der Anteil an gewonnenen Nutzdaten je Aufnahmesitzung zu gering erschien. Daher wurde im weiteren Verlauf die nachfolgend beschriebene Strategie zum Aufbau des Sprachinventars gewählt.

6.1.1. Konzeption und Erstellung der Lesevorlagen

Die neuen Lesevorlagen sollten sich unmittelbar an der Struktur der ausgegebenen Adresseinträge orientieren. Da diese nicht in einen Trägersatzkontext eingebettet werden sollten, sondern die Ausgabe lediglich aus den Namens- und Adressbestandteilen zusammengesetzt wurde, konnten Aufnahmetexte erstellt werden, die außer den häufigen Namensbestandteilen wie ,,Straße" oder ,,Weg" keine Redundanz auf lexikalischer Ebene enthielten. Die Datenbasis bildeten die vom Auftraggeber zur Verfügung gestellten Statistiken über die Häufigkeiten verschiedener Typen in den fünf Eintragsarten ,,Vorname, Nachname, Straße, Hausnummer" und ,,Ort". Als Typen werden hier die unterschiedlichen Inhalte der Felder einer Eintragsart bezeichnet. Jeder Typ konnte aus einem oder mehreren zusammengehörigen Wörtern

Spaltenname	Bedeutung	Beispiele	
id	eindeutige Nr. des Typs	31	31
ix	Index der enth. Wörter	1	2
typ	Eintragsart (v, n, s, o)	v	v
sprache	Herkunftssprache, falls bekannt	de	de
s_text	Text des Eintrags	Karl	Heinz
v_text	Volltext des Typs	Karl-Heinz	Karl-Heinz
k_trans	Transkription	`k"a:6l`	`"haInts`
h_trans	automat. Vorgabetranskription	`k"a6l`	`"haInts`
v_trans	Transkription des Volltextes	`k"a:6l "haInts`	
t_flag	Transkription vorh.?	j	j
vt_flag	Volltexttranskription vorh.?	j	n
pro_flag	progrediente Realisierung aufgenommen?	n	n
med_flag	mediale Realisierung aufgenommen?	j	n
fin_flag	finale Realisierung aufgenommen?	j	n
junk_flag	Aufnahme wiederholen?	n	n
anzahl	Häufigkeit des Volltext-Typs	189837	189837
sex	Geschlecht des Namens	1	1

Tabelle 6.1.1.: Aufbau der Tabelle `haupt` in der Projektdatenbank mit Erläuterungen und Beispieleinträgen.

bestehen. Aus diesen Daten wurde eine MySQL-Projektdatenbank aufgebaut, deren Haupttabelle für jedes Wort eines Typs einen Eintrag mit den in Tabelle 6.1.1 dargestellten Informationen enthielt.

Für jede zu erstellende Lesevorlage wurde von jeder Eintragsart der häufigste bereits vollständig manuell transkribierte aber noch nicht in der entsprechenden Phrasenposition aufgenommene Typ aus der Projektdatenbank ermittelt. Bezogen auf die Tabelle 6.1.1 mussten also die Kriterien erfüllt sein, dass je nach Position in der Zieläußerung eines der Flags pro_flag, med_flag oder fin_flag den Wert „n" für „nein" besaß und das Feld v_text für den ersten Eintrag des Typs (mit $ix = 1$) nicht leer war. Die ausgegebenen Typen wurden folgendermaßen in die Slots des Lesevorlagenschemas eingesetzt:

Vorname Nachname `2,? $p` **Straße** **Hausnummer** `2,? $p` **Ort** `5,.`

alternativ

Vorname Nachname `2,? $p` **Straße** `2,? $p` **Ort** `5,.`

2,? und 5,. stehen hier für eine interne progrediente Phrasengrenze mit steigender bzw. eine finale Phrasengrenze mit fallender Intonation gemäß den in Abschnitt 4.1.1 vorgestellten Konventionen, $p für eine Sprechpause und die fett gedruckten Teile für einen zu füllenden Slot der entsprechenden Eintragsart. Die Zusammensetzung der Lesevorlagen stand dabei in keinem Zusammenhang zu den tatsächlich auftretenden Kombinationen in der Adressdatenbank des Auftraggebers, die für das Projekt auch nicht zur Verfügung gestellt wurde. Eine Lesevorlage konnte so beispielsweise die Form des fiktiven Eintrags

Harald Schmidt 2,? $p **Theaterplatz 23** 2,? $p **Unterkassel** 5,.

annehmen, aber auch wie unten gezeigt aussehen.

Ingrid **Institut für Kommunikationswissenschaften** 2,? $p
Poppelsdorfer Allee 47 2,? $p **Bonn** 5,.

Bei dem zweiten Beispiel entstammt der erste unterstrichene Eintrag der Art Vorname, der zweite ist ein Typ der Art Nachname aus einem oder mehreren Adresseinträgen, die kein ausgefülltes Vornamensfeld besitzen, da es sich um Anschrift und Telefonnummer eines Unternehmens oder einer Institution handelt. Die so erzeugten Lesevorlagen klangen daher mitunter absurd, waren aber im Hinblick auf die Abdeckung der Typen in bestimmten Phrasenpositionen so gewollt. Dabei wurde in Kauf genommen, dass die Nachnamen ohne vorangehende Vornamen im Ursprungskontext nun nicht mehr äußerungsinitial waren, sondern ggf. mit den eingesetzen Vornamen koartikuliert wurden. Der positive Nebeneffekt war, dass auf Phonebene dadurch mehr wortübergreifende Kontexte aufgenommen werden konnten, die durch die starke Phrasierung und Pausensetzung im Lesevorlagenformat ansonsten eher nicht begünstigt wurden.

Da die Straßennamen sowohl mit als auch ohne Hausnummer und damit phrasenmedial oder phrasenfinal geäußert werden konnten, wurden die Lesevorlagen immer paarweise erzeugt und ein gewählter Straßentyp in beide eingesetzt. Die anderen Typen wurden dagegen variiert. Als Beispiel sollen die ersten beiden Korpusäußerungen dienen:

Bernhard Hein 2,? $p **Damm** **1A** 2,? $p **Berlin** 5,.

Ernst Stephan 2,? $p **Damm** 2,? $p **Köln** 5,.

Die Generierung der Lesevorlagen aus der Projektdatenbank wurde durch das Tcl/Tk-Script (Tcl/TK, 2008) gen_lesevorlagen.tcl vorgenommen (Abbildung 6.1.1). Um die Transkriptionen der Typen vor der Aufnahme noch einmal überprüfen zu können, wurde das Skript zur Erzeugung der Lesevorlagen mit einem GUI versehen, das die sukzessive manuelle Auswahl von Vornamen, Namen, Straßen, Hausnummern und Orten erlaubte (Abbildung 6.1.1). Dabei bestand die Möglichkeit, falsch

transkribierte Einträge zur Wiedervorlage für die Transkribenten zu markieren. Die aus den gewählten Typen erzeugten Lesevorlagen wurden in einer weiteren Tabelle der Datenbank gespeichert und dort als noch nicht aufgenommen markiert. Wurden für die Aufnahmen neue Ausdrucke von Lesevorlagen benötigt, konnten über das Perl-Script (The Perl Foundation, 2008) `drucke_lesevorlagen.pl` alle neuen Vorlagen aus der Tabelle ausgelesen, dann in ein Dokument für das Schriftsatzsystem LaTeX (LaTeX, 2008) und schließlich ein PDF-File umgewandelt werden. Die Lesevorlagen enthielten sowohl die orthographische Form als auch die Sollaussprache der Äußerung in boss-sampa.

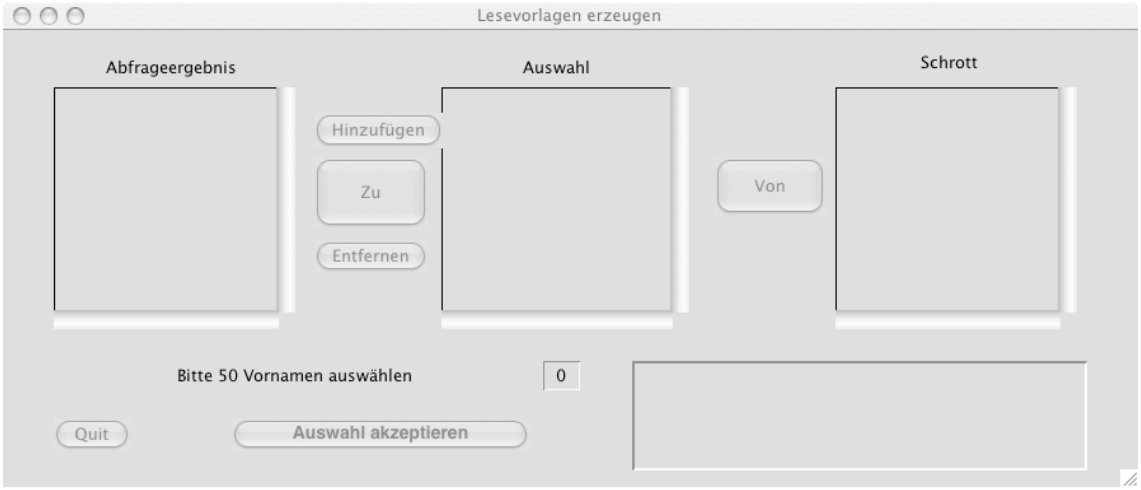

Abbildung 6.1.1.: Screenshot des Programms zur Auswahl von Namen für die Erzeugung der Lesevorlagen.

6.1.2. Aufnahme und Annotation

Die Aufnahme des Korpus fand im Tonstudio des damaligen IKP statt. Die Sprecherin saß dabei mit dem Rücken zur Glasscheibe, die den Aufnahme- vom Kontrollraum trennt, und sprach in Richtung der schallgedämmten Wände, um Reflexionen weitestmöglich zu vermeiden. Vor der Aufnahme wurde überprüft, dass der Abstand zum Mikrofon und die Sitzposition identisch zu den vorangehenden Sitzungen war. Die Aufnahmen wurden nach Möglichkeit seitenweise (bezogen auf den Ausdruck der Lesevorlagen) ohne Unterbrechung durchgeführt und notwendige Wiederholungen nach jeder Seite vorgenommen und notiert. Zwei Aufnahmeleiter begleiteten jede Sitzung, damit die technische Qualität der Aufnahme und die vorlagentreue phonetische Realisierung simultan gewährleistet werden konnten. Weiterhin wurde darauf geachtet, dass die Aufnahmen ungefähr zur jeweils gleichen Zeit am Vormittag durchgeführt wurden. Das Aufnahmesystem war ein PC mit dem Betriebssystem Windows 98 und einer SoundBlaster 16-Bit-Audiokarte, die unmittelbar mit

dem Mikrofon verbunden war. Die Software AudioSliders (Code Sector Inc., 2008) wurde eingesetzt, um die Pegeleinstellungen der Soundkarte über die Sitzungen konstant zu halten. Für die Aufnahme selbst wurde der Audio-Editor CoolEdit 2000 (jetzt Adobe Audition, Adobe Systems Inc., 2008) verwendet. Aufgenommen wurden PCM-kodierte WAV-Dateien mit 16-Bit-Quantisierung und einer Abtastrate von 32 kHz. Zu Beginn der Sitzung musste die Sprecherin einen Referenzsatz äußern, der zur Kontrolle und Anpassung der Sprechgeschwindigkeit und -lautstärke diente. Die Aufnahmen einer Sitzung wurden in einer großen Datei gemeinsam abgespeichert und zusätzlich manuell entsprechend den Notizen zur Aufnahmereihenfolge in Äußerungen von der Größe einer einzelnen Lesevorlage geschnitten. Letztere wurden wiederum für die Annotation und Verwendung in BOSS tiefpassgefiltert, auf 16 kHz Abtastfrequenz konvertiert und als RAW-Dateien ohne Header abgespeichert. Diese Samplingrate war der Standard für die BOSS-Software und erschien für die Anwendung in einem Telefonszenario als ausreichend. Zu allen Dateien wurden auf Basis der orthographischen Einträge in den Lesevorlagen automatisch Wortlabel-Files im BLF-Format (s. Abschnitt 4.1.1) mit äquidistanten Wortgrenzen erstellt. Diese wurden manuell mit dem Programm Wavesurfer an die tatsächlichen Wortgrenzen angepasst und als Eingabe für einen wortbasierten HTK-Segmentierer verwendet. Die daraus generierte Lautsegmentierung wurde für jede Datei wiederum manuell mit Wavesurfer (Sjölander und Beskow, 2000) überprüft und korrigiert. Die Basis der Lautlabel war die in Kapitel 3 vorgestellte Multi-Phon-Definition und das ihr zugrundeliegende boss-sampa. Für die Annotation wurde eine engere Beschreibung gewählt als dort empfohlen ohne jedoch die zur Einheitenauswahl verwendeten kanonischen Etiketten zu entfernen (mit Ausnahme von elidierten Lauten). Die Abweichungen zwischen Realisierung und kanonischer Aussprache wurden in Anlehnung an die Verbmobil-Konventionen folgendermaßen in den Labeldateien festgehalten:

kanonische Realisierung: `a`

abweichende Realisierung: `a-b`

Elision: `a-`

Das mit `b` bezeichnete Etikett wurde bei der Konvertierung der BLF-Dateien in XML als Attribut `TReal` übernommen (s. auch Abschnitt 4.1.2) und auch in der Datebankrepräsentation unter diesem Spaltennamen gespeichert. Die darin enthaltenen Informationen zur Realisierung konnten daher in der Kostenberechnung bei der Einheitenauswahl berücksichtigt werden (Abschnitt 6.3.4).

6.1.3. Zahlenprosodie

Keine Rede war bisher von den für die Telefonauskunft so wichtigen Realisierungen von Zahlen und Ziffern. Da diese die Kerninformation des Dienstes darstellen, muss gerade dort auf maximale Verständlichkeit Wert gelegt werden, so dass nur die Auswahl ganzer Wörter oder Phrasen in Frage kommt. Die Phrasierung der hier hauptsächlich relevanten Telefonnummern und die Entscheidungskriterien für die Aussprache von Zifferngruppen als Zahlen oder Einzelziffern sind komplex und nicht allein von der Anzahl der Ziffern abhängig, sondern auch davon, ob diese eine Zahlenreihe bilden oder sich wiederholen. Sollen die Zahlen im Sinne der reproduktiven Synthese ausgewählt werden, würde die Kombinatorik den Rahmen jedes Korpus sprengen. Hinzu kommen nicht vorhersehbare idiosynkratische Präferenzen des Sprechers zur Gruppierung. Tritt dieser als Hörer in einem Telefonauskunftsszenario mit synthetischer Nummernansage auf, kann nicht erwartet werden, dass dessen Erwartungen exakt erfüllt werden können. Mit einfachen Mitteln ist dennoch eine deutliche Ausgabe durch Reproduktion möglich, die die in vielen Systemen (wie z. B. der T-Net-Box der Deutschen Telekom) eingesetzte monotone Wiedergabe einzelner Ziffern in progredienter Einzelwortaussprache an Natürlichkeit deutlich übertrifft. Dabei werden die Nummern in Blöcke von zwei Ziffern eingeteilt, die am Stück realisiert und dadurch koartikulatorisch verflochten sind. Besteht die Nummer aus einer ungeraden Anzahl von Einheiten, wird die erste Ziffer abgespalten und einzeln ausgegeben. Die Telefonnummer 0228-7356800 würde demnach folgendermaßen phrasiert:

Null Zwei `2,? $p` **Zwei Acht** `2,? $p` **Sieben** `2,?` **Drei Fünf** `2,? $p`
Sechs Acht `2,? $p` **Null Null** `5,.`

Dieses Verfahren wurde für die hier dargestellte Telefonauskunftssynthese entwickelt und realisiert. Die Korpusgrundlage, die dies ermöglichte, bildeten die Aufnahmen der 100 Paare „Null Null" bis „Neun Neun" und der Ziffern Null bis Neun, alle jeweils in progredienter und phrasenfinaler Realisierung. Parallel veröffentlichten (Baumann und Trouvain, 2001) ein ähnliches Konzept.

Analog wurde mit der Aufnahme und Ausgabe der Realisierung von Buchstaben verfahren, um eine ebenso verständliche Buchstabierung von Wörtern zu ermöglichen. Sowohl die Zahlen- als auch die Buchstabenaufnahmen wurden nicht phonetisch annotiert, sondern als dedizierte Einheiten auf Wortebene segmentiert und orthographisch bzw. als Ziffern mit dem Präfix § etikettiert (`§A`, `§1` usw.).

Neben den bisher erwähnten, häufigen Typen von Lesevorlagen wurden weitere Texte zur Ergänzung des Inventars aufgenommen. Dazu gehören z. B. die Realisierungen seltener Multi- und Xenophone. Eine vollständige Übersicht liefert Anhang E.

6.1.4. Pruning

Bei der informellen Überprüfung der Synthesequalität ergab sich gegen Ende des Projektes der Eindruck, dass ein größerer Anteil von Segmenten im Korpus zu schnell und reduziert realisiert worden war. Aus diesem Grunde wurden die Lautdauern in der Datenbank statistisch analysiert und alle Instanzen entfernt, deren Dauer unterhalb des ersten Quartils der Dauer des Lauttyps lag. Tabelle 6.1.2 zeigt ein Beispiel für das Multiphon `hI`.

Einheit	0%	25%	50%	75%	100%
`hI`	56, 0600	59, 5175	63, 5950	106, 4400	135, 1900

Tabelle 6.1.2.: Quartile der Lautdauer (ms) für den Multiphon-Typ `hI`.

Ebenso wurden alle Silben und Wörter entfernt, deren Dauer geringer als die aufsummierten ersten Quartile der enthaltenen Einheitentypen war. In einem weiteren Schritt zur Verbesserung der Qualität wurden alle unterschiedlichen `TReal`-Einträge auf Phonebene betrachtet und die Instanzen all jener Typen aus der Datenbank entfernt, die auf eine sehr reduzierte Realisierung oder einen Annotationsfehler hindeuteten. Zudem wurden alle Funktionswörter und deren Bestandteile aus den besonders schnell gelesenen Lesetexten zur phonetischen Abdeckung herausgenommen, um zu verhindern, dass diese zur Synthese von Namen verwendet werden. Weitere auffällige Wörter wurden in Listen gesammelt und ebenfalls entfert. Die Sprachdaten und Annotationen wurden dabei nicht physisch aus dem Korpus gelöscht, sondern nur über eine Spalte `Align` als nicht zum Gebrauch vorgesehen markiert. Dieses Flag, das die Werte `Auto` und `Manual` annehmen kann, diente während der Aufbauphase des Korpus dazu, in der Einheitenauswahl eine Präferenz der bereits manuell korrigierten Einheiten zu verwirklichen. Dazu wurde die Vorauswahl so gestaltet, dass zunächst nur manuell segmentierte Einheiten abgefragt wurden und nur im Fall einer leeren Treffermenge zusätzlich nach den automatisch segmentierten gesucht wurde. Nach Abschluss der Aufnahmen war dies nicht mehr notwendig und das Etikett `Auto` wurde genutzt, um durch das Pruning ausgeschlossene Einheiten zu identifizieren.

6.1.5. Quantitative Analyse

Das abgeschlossene Korpus umfasst 4.138 Dateien, die in der beschriebenen Kodierung 624 MB belegen. Zusammengenommen beläuft sich die Länge des Korpus auf fünf Stunden und vierzig Minuten. Die Nettolänge ohne entfernte Einheiten und

Pausen liegt bei 201 Minuten. Tabelle 6.1.3 gibt einen Überblick über die Zahl der verschiedenen Einheitentypen vor und nach dem Pruning (ohne Pausen).

	Wörter	Silben	Phone
vorher	31.195	61.217	138.525
nachher	25.071	45.872	94.688
entfernt	6.124	15.345	43.837

Tabelle 6.1.3.: Anzahl der verschiedenen Einheitentypen vor und nach dem Pruning.

Um zu ermitteln, wie sich das Korpus zur angestrebten Domäne ,,Telefonbuch" verhält, wurde die Abdeckung der Volleinträge für jede Eintragsart durch in der richtigen Phrasenposition aufgenommene zusammengesetzte Worteinheiten analysiert. Für jeden in der Projektdatenbank bereits transkribierten Volleintragstyp wurden alle konstituierenden Worteinheiten gesucht. Dabei wurde für das letzte Wort des Volleintrags die Phrasenposition der Eintragsart im angestrebten Ausgabeformat (s.o.) angenommen, für alle vorangehenden die mediale Position. Entscheidend war nicht, ob die Wörter zusammenhängend geäußert wurden, sondern nur dass die Phrasierung der Einzelwörter der Zielvorgabe entsprach. Nur wenn alle Wörter eines Volleintragstyps in dieser Hinsicht prosodisch adäquat waren, wurde der Volleintrag als sicher mit hoher Qualität synthetisierbar bewertet. Das Resultat dieser Analyse gibt also die untere Grenze der Abdeckung der Eintragstypen durch eine Wortsynthese mit passender globaler Intonationskontur an. Da aber möglicherweise für viele weitere Einträge gut geeignete Silben- und Phonkombinationen möglich sind, liegt die Anzahl qualitativ hochwertiger Volltextsynthesen vermutlich erheblich höher. Andererseits konnte die interne Phrasierung längerer Volltexteinträge (bspw. ,,A. A. A. A. Schlüsseldienst") nicht sicher vorhergesagt und daher in der Analyse nicht berücksichtigt werden, so dass hier ein gewisses Fehlerpotenzial der Prognose liegt.

Die Anzahlen der Vorkommen aller durch Worteinheiten abgedeckten Eintragstypen in der Telefondatenbank wurden aufaddiert und in Beziehung zur Gesamtzahl von Eintragsinstanzen gesetzt. Darauf aufbauend zeigen die in Abbildung 6.1.2 dargestellten prozentualen Werte für jede Eintragsart, wie viele der tatsächlichen Instanzen eines Typs im Telefonbuch auf diese Weise synthetisiert werden können. Die Zahlen stimmen optimistisch, obwohl die in Abbildung 6.1.3 dargestellte Abdeckung der reinen Typen durch Worteinheiten wenig beeindruckend ist. Eine Auflistung der absoluten Werte findet sich in Tabelle E.2.1 im Anhang.

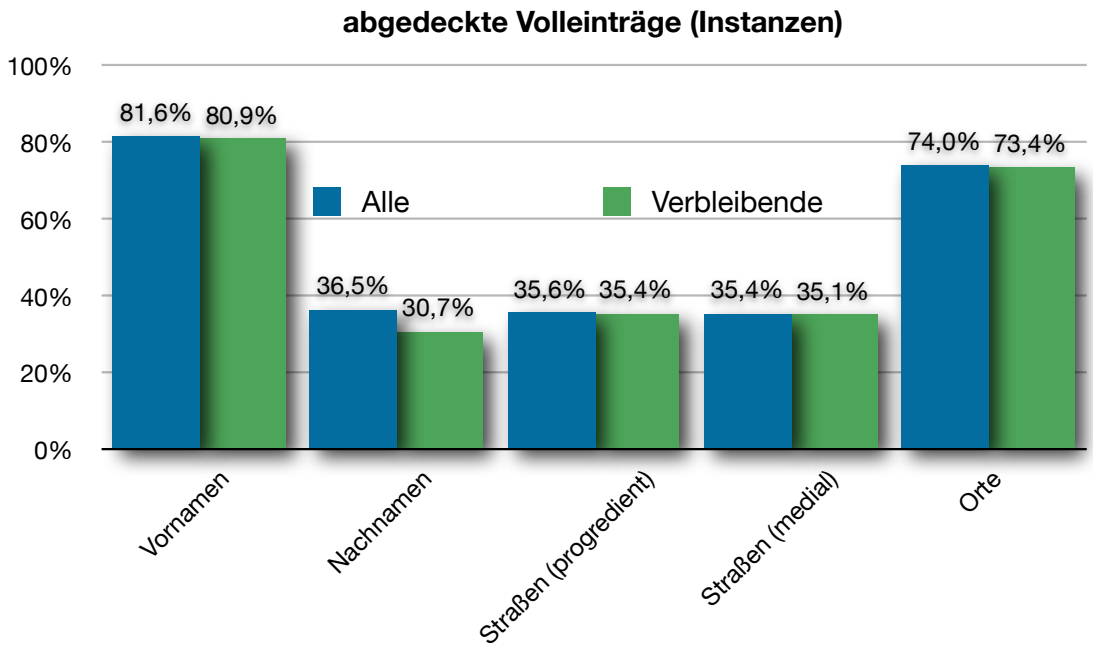

Abbildung 6.1.2.: Abdeckung der nach Eintragsarten sortierten Instanzen der Volleinträge in der Telefondatenbank durch die Worteinheiten des Korpus vor und nach dem Pruning (Alle/Verbleibende) in Prozent. Die Absolutwerte finden sich im Anhang in Tabelle E.2.1.

6.2. Aufbau eines Aussprachewörterbuchs

Die klickTel-Synthese verfügt über ein großes Aussprachewörterbuch mit über 750.000 Einträgen, das aus den im Laufe des Projektes angefertigten Namenstranskriptionen und dem *Bo*nn *M*achine-Readable *P*ronunciation Dictionary, kurz BOMP (Portele et al., 1995), zusammengesetzt wurde. Da aus Gründen der Laufzeiteffizienz verschiedene Versionen der Orthographie eines Eintrags schon im Lexikon kodiert sind, liegt die Zahl der tatsächlich transkribierten Wörter deutlich darunter. In den folgenden Abschnitten werden zunächst die Transkription von Eigennamen problematisiert und die für das Projekt getroffenen Konventionen zu deren Erstellung geschildert, bevor die einzelnen Komponenten des Wörterbuchs näher beschrieben werden.

6.2.1. Aussprache von Eigennamen

Es ist aus verschiedenen Gründen oft schwierig, die Lautung eines unbekannten Namens zu bestimmen. Zum einen ist aus historischen Gründen die Graphem-Phonem-Beziehung bei Eigennamen noch weniger konsistent als in den erst neuzeitlich geschaffenen orthographischen Konventionen, so dass z. B. die Vokalquantitäten nicht

eindeutig bestimmt werden können. Dies kann der Fall sein, wenn der Name über die Jahrhunderte Reduktionsprozessen unterzogen war und in der reduzierten Fassung lexikalisiert wurde. Dann kann etwa von der Abwesenheit von Doppelkonsonanten nicht notwendigerweise auf einen vorangehenden Langvokal geschlossen werden. Der Ortsname ,,Drabenderhöhe" ist mit seiner Aussprache [dʁamdɐhøːə] ein extremes Beispiel solcher Verschleifungen. Ein anderes Problem ist die regional-dialektale Interpretation von Graphemfolgen, wie z. B. von ⟨st⟩ als [ʃt] in ,,Konstanz", ⟨ui⟩ als [yː] in ,,Duisburg" und ⟨oi⟩ als [oː] in ,,Troisdorf". Selbst wenn diese Regeln bekannt sind, steht der Sprecher oder Transkribent vor dem Problem, ob er in seiner Realisierung oder Verschriftung eine vermeintlich standarddeutsche Lesart wählen soll (die für Eigennamen nicht konventionalisiert ist) oder die regionale Variante wählt. Gleiches gilt für die obengenannten Reduktionsformen und aus der Schreibung abgeleitete ,,orthophone" Varianten (nach lokaler Auffassung gilt: *[dʁabəndɐhøːə]!). Dies ist sicherlich keine Option für die bekannteren der genannten Beispiele, ermöglicht aber bei unbekannten analogen Fällen einige Freiheitsgrade, so dass die Realisierung immer eine Mischung aus dem Weltwissen und subjektiven Empfinden eines bestimmten Sprechers ist. Es existiert also nicht *die eine* Aussprache eines Namens, was sich auch an den verschiedenen Aussprachevarianten von Familiennamen demonstrieren lässt. Eine dem Autor bekannte Kölner Familie Hucklenbroich realisiert ihren Namen beispielsweise [hʊklənbʁɔɪç], da das sogenannte rheinische Dehnungs-i in Köln fast

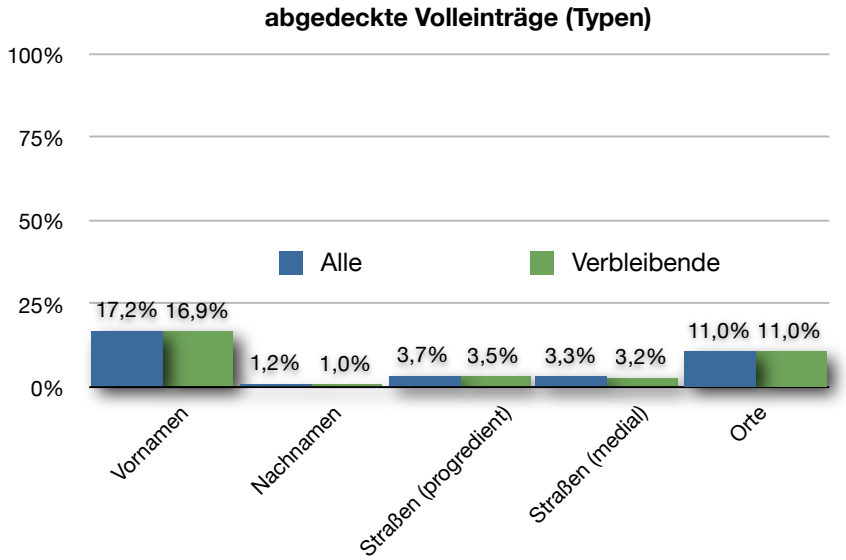

Abbildung 6.1.3.: Abdeckung der nach Eintragsarten sortierten Typen der Volleinträge in der Telefondatenbank durch die Worteinheiten des Korpus vor und nach dem Pruning (Alle/Verbleibende) in Prozent. Die Absolutwerte finden sich im Anhang in Tabelle E.2.1.

gänzlich unbekannt ist. Eine seit mehreren Generationen in Grevenbroich ansässige (fiktive) Familie würde hier vermutlich zu einer anderen Entscheidung kommen.

Auch die lexikalisch akzentuierte Silbe ist nicht immer festzustellen, insbesondere bei Ortsnamen. Eine informelle Beobachtung ist, dass bei polymorphemischen Namen mit mehreren Stämmen einheimische Sprecher oft einen Standard-Primärakzent verwenden, während Außenstehende einen Sekundärakzent bevorzugen (**Eus**kirchen vs. Eus**kir**chen und **Wald**bröl vs. Wald**bröl**). Dies liegt möglicherweise an einer stärkeren Lexikalisierung des Namens durch die erste Gruppe gegenüber einer morphologischen Dekomposition durch die auswärtigen Sprecher.

Eine noch größere Schwierigkeit stellt die Transkription fremdsprachlicher Namen dar. Während im Deutschen nur die Graphem-Phonem-Beziehungen und die Prosodie kritisch sind, müssen bei ausländischen Namen auch die Phonem-Phonem-Relationen beachtet werden. Ein Sprecher des Deutschen wird für seine Aussprache eines deutschen Namens in aller Regel das Lautinventar des Standarddeutschen oder seines Regio- bzw. Dialektes verwenden. An die Aussprache ausländischer Namen wird jedoch, wenigstens für Namen, die einer aus dem Schulunterricht bekannten und stark verbreiteten Sprache wie dem Englischen oder Französischen entstammen, oft die Erwartung geknüpft, dass zumindest ein Teil der Lautung phonetisch der Ursprungssprache entspricht. Die Erwartungshaltung ist damit nicht einfach von einer binären Entscheidung zwischen vollständig authentischer oder komplett eingedeutschter (oder allgemeiner: „nativisierter") Aussprache abhängig, sondern bezieht sich meist nur auf bestimmte Laute (Xenophone) und phonotaktische Phänomene.

6.2.2. Konventionen zur Transkription

Um eine möglichst konsistente Transkription durch verschiedene Transkribenten zu ermöglichen, wurden Richtlinien für die Behandlung deutscher und ausländischer Namen verfasst. Die Basis der Transkription war, in Übereinstimmung mit der Annotation der Datenbank, von Anfang an die boss-sampa-Spezifikation inklusive Phoxsy-Multiphonen. Da die Lautsymbole einer Transkription im Lexikon nicht durch Leerzeichen oder andere Markierungen getrennt werden, liegt die Unterscheidung zwischen reinem boss-sampa und boss-sampa mit Phoxsy nur in der Platzierung der Wortakzentsymbole. Boss-sampa verlangt eine Annotation unmittelbar vor dem Silbenkern. Ist dieser aber Teil eines Multiphons, so wandert das Symbol vor den Anlaut dieser Einheit:

`.fl"o:` → `.f"lo:`

Das Symbolinventar, das für die Transkriptionen zur Verfügung stand, umfasste 410 Phon- und Multiphon-Einheiten. Dies entspricht der in Tabelle A.2.1 im Anhang

gelisteten Menge von Phoxsy-Symbolen, aber beinhaltet nicht die für die Korpusannotation zusätzlich benötigten Einheiten zur Markierung von Pausen (`$p`), nicht verwendbaren Einheiten (`$j`) oder die mit § präfigierten dedizierten Alphabet- und Ziffernkombinationen. Bei den zusammengesetzten Lauten wurden nicht alle Kombinationen zugelassen, da einige Phone nur für Transkriptionen bestimmter Sprachen verwendet werden; so wird englisches [ɹ] (`R`) bspw. nicht mit Nasalvokalen kombiniert zugelassen und taucht daher im Lautset nicht auf. Im Folgenden werden die wichtigsten Regeln für die Annotation im klickTel-Projekt dargestellt.

6.2.2.1. Wortakzent

Bei Ortsnamen und anderen ,,Pseudo-Komposita" (z. B. ,,Kirchenthumbach") kann die Platzierung des Wortakzentes nicht eindeutig bestimmbar sein. In diesen Fällen sollte standardmäßig die erste akzentuierbare Silbe des Wortstamms als Wortakzentträger markiert werden:

Wiesenfelden `"vi:.z@n.f%El.d@n`

Nebenakzente sollten nur für solche Morpheme vergeben werden, die auch synchron noch erklärbar sind, dies trifft z.B. zu für ⟨feld⟩, ⟨weg⟩, ⟨tal⟩, ⟨kirch⟩ oder ⟨mann⟩ bei Personennamen. Grundsätzlich sollten sich die Transkribenten dabei immer auf ihre muttersprachliche Intuition verlassen.

Durch Bindestrich verbundene Orts-, Straßen und Personennamen wurden als mehrere Wörter betrachtet und erhielten zwei Primärakzente. Dasselbe gilt für Namenszusätze wie ,,Bad" in ,,Bad Neuenahr".

Villingen-Schwenningen `f"I.lI.N@n S"vE.nI.N@n`

6.2.2.2. Vokalqualität und -quantität

Zur Verbesserung der Verständlichkeit wurde bei der Transkription von Vokalen in Eigennamen immer die gespannte Version bevorzugt, wenn sowohl die gespannte als auch die ungespannte Variante plausibel erschienen. So z.B. auch bei der Graphemfolge ⟨ael⟩:

Michael `m"I.Ca:.?e:l`

Aus den gleichen Gründen wurde die Graphemfolge ⟨ar⟩ in Namen stets als `a:6` transkribiert, es sei denn, sie wurde von einem weiteren ⟨r⟩ gefolgt.

Wie im obigen Beispiel ,,Michael" wurde der Hiat(us), also die Verbindung von zwei Vokalen in aufeinanderfolgenden Silben ohne abgrenzenden Glottalverschluss, wo möglich vermieden, aber explizit erlaubt, wo er auch im Deutschen unplausibel ist:

Lea `l"e:.a:`

6.2.2.3. Behandlung von Silbenreduktion

Für ein Syntheseinventar mit deutlicher Lesesprache sind reduzierte Formen wie `"li:l.j@` für „Lilie" und `"ju:l.ja:` für „Julia" ungeeignet. Daher sollte in der Transkription die unreduzierte Variante verwendet werden, also `"li:.li:.@` und `"ju:.li:.a:`.

6.2.2.4. Fremdsprachliche Laute und Eigennamen

Um den Phonem-Phonem-Beziehungen zwischen fremdsprachlichen Eigennamen und ihren deutschen Realisierungen besser gerecht werden zu können, wurden einige Xenophone in Form von Phon- und Multiphon-Einheiten aufgenommen. Es handelt sich dabei, bis auf die üblicherweise ohnehin zum deutschen Inventar zugerechneten französischstämmigen Nasalvokale, durchweg um englische Laute. Dabei wurden nur solche Laute ausgewählt, von denen angenommen werden konnte, dass sie von einer Vielzahl der Hörer in der Aussprache erwartet werden. Diese Laute durften auch für die Transkription von Eigennamen verwendet werden, die nicht aus dem Französischen oder Englischen stammten. Die folgenden englischen Xenophone standen für die Transkription zur Verfügung:

[θ] `T`, [ð] `D`, [eɪ] `eI`, [oʊ] `oU`, [ɘː] `9:`, [ɔː] `O:`, [w] `w`, [ɹ] `R`

Bis auf die Verwendung von `9:` zur Transkription des Englischen[1] konnte Abresch (2007) anhand von Präferenztests zeigen, dass die Reduktion auf dieses Inventar der Erwartungshaltung einer Mehrheit der Hörer aus verschiedenen Altersstufen und soziologischen Strukturen entspricht. Für `9:` empfiehlt sie allerdings die Substitution durch `96`. Die in beiden Varianten implizite nicht-rhotische Realisierung des Koda-/r/, wie im britischen Standard, wurde jedoch bestätigt. Bei den Diphthongen war hingegen die Bevorzugung der amerikanischen Aussprache [oʊ] gegenüber britischem [əʊ] offenbar eine sinnvolle Annahme. Weitere Bestätigung im Bereich der Phonotaktik gab es für die im klickTel-Lexikon universelle Anwendung der Auslautverhärtung und die Lockerung des Verbots von [s] im Onset. Generell stellte Abresch eine Zurückhaltung gegenüber englischen Vokalen, sowohl in der Produktion als auch in der Akzeptanz bei der Perzeption fest. Daher stellte sich auch die Entscheidung, diese nach phonetischer Ähnlichkeit[2] auf deutsche Laute abzubilden, als richtig heraus:

[1] Das Symbol 9: wurde auch zur Transkription von frz. `oeuvre` benutzt.
[2] — aus Sicht des deutschen Muttersprachlers —

cat `k"Et` statt `k"{t` [ˈkæt]

Für das Französische wurden nur die drei Nasalvokale [ɔ̃] `O~`, [ɛ̃] `E~`, [ɑ̃] `A~` berücksichtigt, da ein Zusammenfall des gerundeten halb-geöffneten Nasalvokals [œ̃] mit dem ungerundeten Pendant [ɛ̃] zu beobachten ist. Außerdem wurde meist die stark verbreitete nativisierte Aussprache [ɔ̃] für [ɑ̃] transkribiert, so dass in kommenden Phoxsy-Versionen möglicherweise auch auf das Symbol `A~` verzichtet werden kann.

Restaurant `rEs.to:."rO~` statt `rEs.to:."rA~`

Kurzzeitig wurde auch der Laut [ɥ] als `H` ins Inventar aufgenommen, später aus Gründen der Kontextökonomie aber mit [w] `[w]` zusammengefasst:

Dubois `dy:.b"wa` statt `dy:.b"Ha`

Generell wurden Xenophone, die nicht ins Lautset aufgenommen wurden, nach phonetischer Ähnlichkeit eingedeutscht (s. o.) oder aufgrund der Orthographie in deutsche Laute überführt. Im Deutschen nicht mögliche Konsonantencluster wurden in einigen Fällen an die deutsche Phonotaktik angepasst. War die Stellung des Wortakzents eines fremdsprachigen Namens nicht eindeutig bestimmbar, wurde sie eingedeutscht. Auch bei türkischen Namen wurde die deutsche Akzentgebung bevorzugt:

Murat `m"U.rat` statt `mU."rat`

In englischen Wörtern wurden die Schwas in unbetonten Silben wenn möglich durch Vollvokale ersetzt, und in der Endsilbe ⟨er⟩ wurde `6` anstatt `@` transkribiert. Auch in Sprachen, in denen ein vokalischer Onset ohne Glottalverschluss möglich ist (wie zum Beispiel im Englischen), wurde dieser transkribiert. Kam ein Name in verschiedenen Sprachen vor, wurde nach der Mehrheit der Einträge im Telefonbuch entschieden; im Zweifelsfall wurde immer eine deutsche Aussprache bevorzugt.

Weitere Richtlinien, die eher die Graphem-Phonem-Beziehung in verschiedenen Herkunftssprachen betreffen, wurden von Abresch in einem projektinternen Handbuch zusammengefasst. Die Darstellung der Regeln für die dort behandelten Sprachen Türkisch, Englisch, Französisch, Russisch, Italienisch, Spanisch, Serbokroatisch, Slowenisch, Tschechisch, Polnisch, Rumänisch, Persisch, Arabisch, Griechisch und Niederländisch würde allerdings den Rahmen dieser Arbeit sprengen.

Um die Aussprache von besonders schwierigen Fällen aus der Liste der deutschen Ortsnamen zu klären, wurden teilweise auch die Rathäuser der entsprechenden Gemeinden kontaktiert und die Mitarbeiter befragt.

Nicht alle Richtlinien wurden unmittelbar am Anfang des Projektes erarbeitet, so dass das Wörterbuch trotz großflächiger Überprüfung am Projektende einige Inkonsistenzen aufwies und aufweisen musste, wenn die betreffenden Wörter bereits in der festgelegten Form aufgenommen worden waren.

6.2.3. Transkription der Namensdatenbank

Bei der Erstellung der oben beschriebenen Projektdatenbank wurden alle Einträge mit einer vorläufigen automatischen Transkription versehen, die mittels HADIFIX (Portele et al., 1995) generiert wurde. Diese Transkriptionen wurden in das Format ,,boss-sampa mit Phoxsy" konvertiert und bildeten so die Grundlage für die manuell erstellten segmentalen Beschreibungen. Um eine Bearbeitung der Einträge nach Häufigkeit und durch verschiedene Transkribenten zu gewährleisten, und um Mehrfachtranskriptionen zu vermeiden, wurde eine eigene Software mit einer grafischen Oberfläche zur Korrektur der lautlichen Verschriftungen geschaffen.

Das Tcl/Tk-Script klinton (*kli*ckTel *n*ifty *t*ranscription *o*f *n*ames) wurde im klickTel-Projekt genutzt, um die Transkription der Datenbankeinträge vorzunehmen. Die Oberfläche dieses Programms ist in Abbildung 6.2.1 dargestellt.

Das Script sucht sukzessive den häufigsten noch nicht vollständig transkribierten Typ einer Eintragsart. Dabei wird zwischen Vornamen, Namen, Straßen und Orten alterniert. Im Feld `Volltext` der Abbildung ist der komplette Eintragstyp zu sehen, der hier nur aus einem Wort (,,Sub-" oder ,,Teileintrag") besteht (im Fall eines Doppelnamens wie „Hans-Peter" wären es zwei Elemente). Darunter steht die Volltranskription; noch nicht transkribierte Elemente werden mit `<fehlt>` markiert. Das Feld Subeintrag zeigt nacheinander die zu transkribierenden Teileinträge zu einem Volleintrag, z. B.:

v_text	v_trans	id	ix	typ	sprache	s_text	k_trans
Hans-Peter	"hans p"e:.t6	1212	1	v	de	Hans	"hans
Hans-Peter	"hans p"e:.t6	1212	2	v	de	Peter	p"e:.t6

Mit dem Knopf `Übernehmen` kann die maschinell erzeugte HADIFIX-Transkription als Vorgabe in das Transkriptionsfeld `k_trans` („klickTel-Transkription") übernommen und dort korrigiert werden. Mit einem Klick auf den `Check`-Button wird die Lautschriftumsetzung anhand regulärer Ausdrücke phonotaktisch überprüft und die Position von Wortakzentzeichen verändert, falls diese sich innerhalb von Multiphonen befinden sollten. Durch Anklicken von `OK` wird die Transkription nach einer nochmaligen Überprüfung für den Subeintrag übernommen und außerdem in der Transkription des Volleintrages, sowohl in der Datenbank als auch im Anzeigefeld, ergänzt. Gleichzeitig werden auch die homographen Subeinträge anderer Volleinträge mit dieser Transkription gefüllt, wenn nicht bereits ein Lautschrifteintrag existiert. Dabei wird zunächst von einer vollständigen Homonymie auch zwischen verschiedenen Eintragsarten ausgegangen. Dieser Ablauf wiederholt sich, bis alle Teileinträge transkribiert wurden und die Volltranskription vollständig ist. Durch Druck auf den

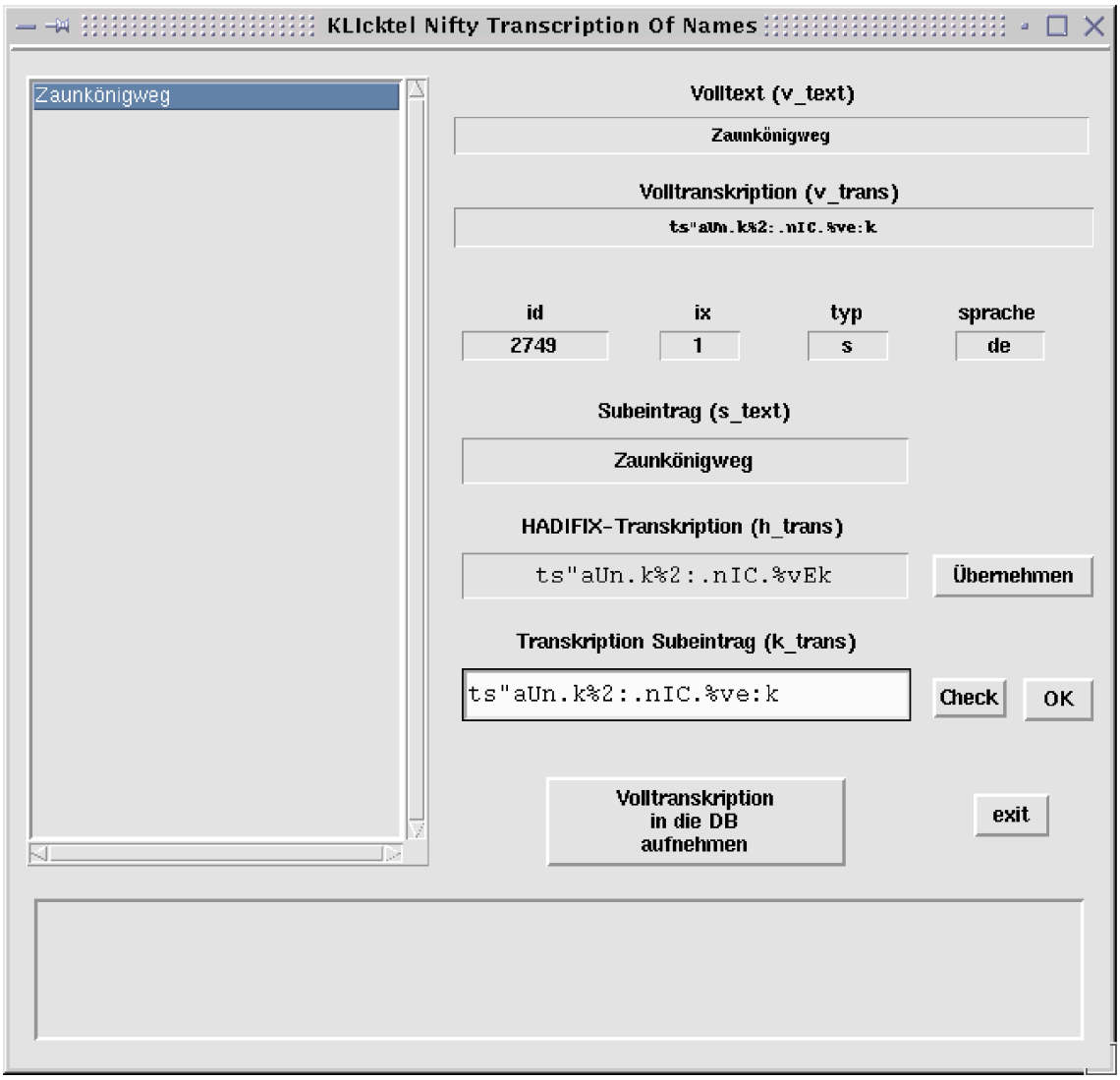

Abbildung 6.2.1.: Grafische Oberfläche des Programms zur Erstellung von Namenstranskriptionen.

entsprechenden Button kann diese dann in die Datenbank aufgenommen werden. Erst dann wird der Volleintrag als komplett verlautschriftlicht markiert (`vt_flag`) und kann durch das oben vorgestellte Programm zur Erstellung von Lesevorlagen verwendet werden. Andere Einträge, deren Bestandteile dabei *en passant* mittranskribiert wurden, werden nicht als fertig markiert und müssen somit vor Verwendung als Lesevorlage erst durch Wiedervorlage in klinton überprüft werden. Dabei können dann auch nicht homophone Einträge vorgenommen werden (z. B. für den engl. Vornamen und den dt. Nachnamen „George"). Die letzte Kontrollinstanz für die Transkriptionen stellt der bereits vorgestellte Mechanismus zum Aussortieren von Volleinträgen in der Auswahl von Lesevorlagen.

6.2.4. Bonn Machine-Readable Pronunciation Dictionary

BOMP wurde am IKP für das konkatenative Synthesesystem HADIFIX entwickelt. Es ist das Produkt zahlreicher Arbeitsstunden wissenschaftlicher Mitarbeiter und studentischer Hilfskräfte, die das Wörterbuch durch manuelle Korrektur von HADIFIX-transkribierten Wörtern, beispielsweise aus Nachrichtentexten, über Jahre stetig erweiterten. Für die Verwendung mit HADIFIX wurde das Lexikon auf die Wörter reduziert, deren Transkriptionen nicht korrekt durch die regelbasierten Algorithmen des Systems erzeugt werden konnten. BOMP bestand in dieser Version mehr oder weniger aus Grundformen der Wörter, deren phonetischer Beschreibung in SAMPA und Angaben zur Wortart. Da aber auch der Einsatz im BOSS-System und in der deutschen Festival-Version des Instituts für maschinelle Sprachverarbeitung in Stuttgart (Möhler et al., 2007) angestrebt wurde, sollte das Lexikon um Vollformen erweitert werden. Im Jahre 2001 wurden daher alle Wörter in der reduzierten BOMP-Version abhängig von ihrer Wortart um verschiedene Kombinationen von Derivations- und Flexionsmorphemen erweitert und mit Hilfe von HADIFIX in Lautschrift umgesetzt. Das Resultat enthielt damit eine große Menge Nonsenswörter, die von Möhler durch Vergleich mit den orthographischen Einträgen anderer Lexika herausgefiltert wurden. Für BOSS wurde das Resultat nach den Richtlinien von boss-sampa und Phoxsy umgesetzt. Die für die Telefonauskunft genutzte BOMP-Version bestand aus 141.854 unterschiedlichen Einträgen.

6.2.5. Erstellung des Gesamtlexikons für die Laufzeit-Synthese

Aus den Transkriptionen in der Projektdatenbank wurden mehrere nach Eintragsart aufgeteilte Aussprachelexika erstellt. Bei der automatischen Extraktion und Zusammenführung dieser Teile mit BOMP und einem separat erstellten Lexikon von 14.000 Firmennamen zur Erweiterung der Abdeckung wurde ein System von Regeln angewendet, das beispielsweise dafür Sorge trug, dass die im Projekt angefertigten Transkriptionen nicht durch BOMP-Einträge überschrieben wurden. Nicht-homophone Homographen wurden nach Namensart in separate Lexika zur Verwendung in der Textvorverarbeitung der Synthese eingetragen. Zu den eindeutig abgekürzten Straßennamen, z. B. mit der Endung ⟨str.⟩ wurden zusätzlich alle expandierten Formen gespeichert. Ebenso wurden umgekehrt alle auf ⟨straße⟩ oder ⟨strasse⟩ endenden Namen dieser Art in die abgekürzte Form überführt. Von allen orthographischen Einträgen wurde eine Version in Groß- und Kleinschreibung sowie vollständiger Kapitalisierung geschaffen, da die Einträge in der Telefondatenbank in dieser Hinsicht uneinheitlich waren. Zum Schluss wurden alle deutschen Umlaute noch in ihre alternativen Schreibungen umgesetzt und alle (wortintern homogenen) Varianten

gespeichert. Bei allen Transformationsschritten wurde garantiert, dass bereits bestehende Transkriptionen zu einer orthographischen Variante nicht überschrieben wurden. Das resultierende Lexikon umfasst 751.994 Einträge. Durch den anschließenden Import der später entstandenen Firmennamen- und der relevanten BOMP-Transkriptionen in die Projektdatenbank konnte rückblickend nicht mehr ermittelt werden, wie viele der Volleinträge manuell über klinton erzeugt wurden. In (Abresch und Breuer, 2004) wird diese Zahl allerdings mit 45.000 angegeben.

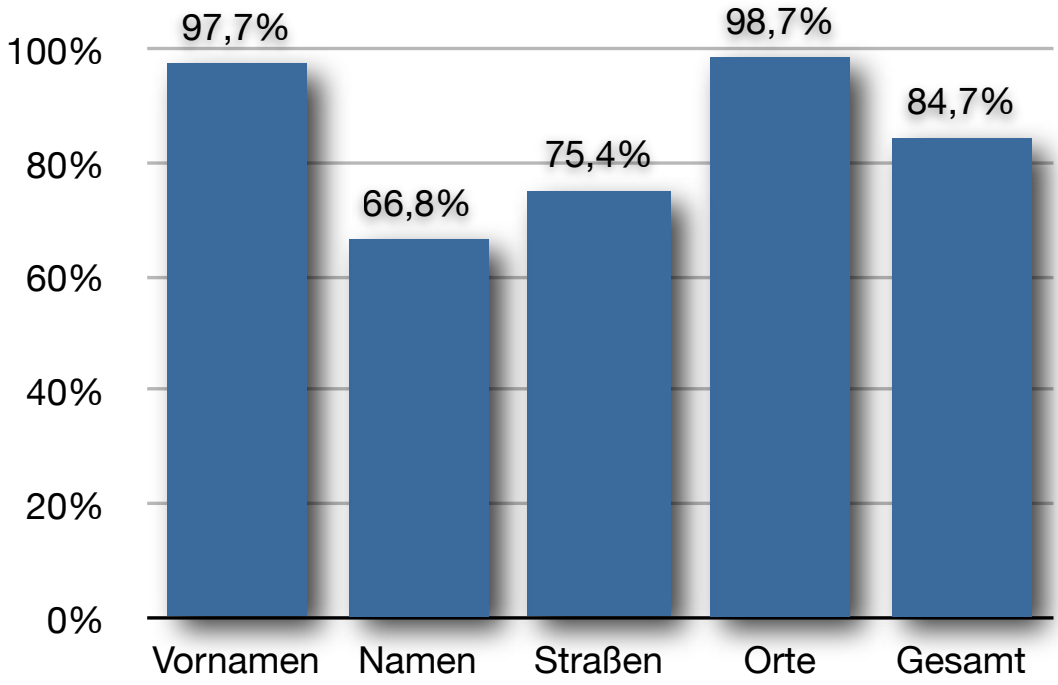

Abbildung 6.2.2.: Abdeckung der nach Eintragsarten sortierten Instanzen der Volleinträge im Telefonbuch durch das Aussprachelexikon und Gesamtabdeckung.

Die auf diese Weise erstellten Transkriptionen wurden, ähnlich wie die aufgenommenen Worteinheiten in Abschnitt 6.1.5 in Beziehung zur Anzahl der Instanzen der durch sie repräsentierten Volleinträge gesetzt. Wie aus Abbildung 6.2.2 ersichtlich, ergibt sich für die Vor- und Ortsnamen schon eine fast vollständige Abdeckung und für die Instanzen aller Arten und Typen immerhin ein Anteil von 84,7%, obwohl nur knapp zehn Prozent der fast 3,5 Millionen Eintragstypen in Lautschrift vorliegen.

Insgesamt werden über 105.000 verschiedene Wörter (Subeinträge) und dadurch 347.704 unterschiedliche Volleinträge über alle Eintragsarten von den Vornamen bis zu den Orten von den erstellten und BOMP entnommenen Transkriptionen abgedeckt. Ausführliche Statistiken zu den Einträgen und deren Transkriptionen finden sich in Tabelle E.3.1 im Anhang.

6.3. Laufzeitkomponenten der Synthese

Das klickTel-Synthesesystem beruht auf der BOSS-Version II, enthält aber schon einige der in Kapitel 5 geschilderten Erweiterungen und eine speziell auf die Aufgaben der Adress- und Nummernwiedergabe zugeschnittene Vorverarbeitung, die das Prinzip des Content-to-Speech verfolgt. Anfänglich war diese Komponente gemäß der in BOSS empfohlenen Aufteilung in einem separaten CTS-Client untergebracht, wurde aber auf Wunsch des Auftraggebers später in den Server integriert. Passend zu den Abbildungen 4.2.1 und 5.5.1 zeigt die Darstellung 6.3.1 die Architektur der klickTel-Version von BOSS.

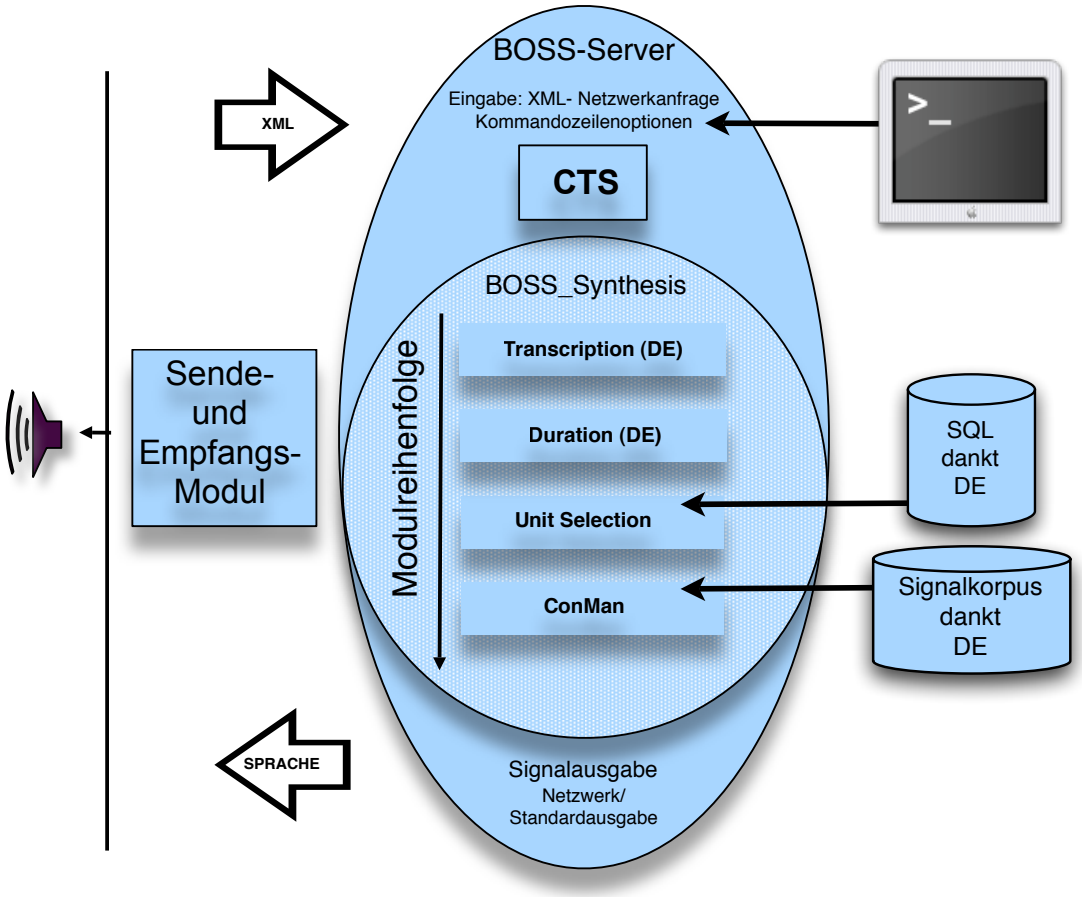

Abbildung 6.3.1.: Architektur des Laufzeit-Systems der klickTel-Synthese.

6.3.1. Textnormalisierung

Anders als in einem reinen Text-to-speech-System existieren bei der Synthese von Adressen a priori Metainformationen zu den Wortbestandteilen und der Rahmenstruktur der Zieläußerung, so dass schon von Concept-to-Speech gesprochen werden kann. Diese Informationen beschränken sich allerdings auf die Angabe, welcher Gruppe von Namen (Vorname, Nachname usw.) oder Zifferngruppen (Haus- oder Telefonnr.) die einzelnen Bestandteile zugehören und wie diese global phrasiert werden sollen. Da die einzelnen Einträge zu den Namen Abkürzungen, Zahlen, Sonderzeichen, Interpunktion etc. enthalten und in überwiegend kurze, unvollständige syntaktische Strukturen eingebettet sein können, besitzt die Aufgabe der Textvorverarbeitung für das Auskunftsszenario die vollständige Komplexität des Text-to-Speech, und dies sogar mit einem Schwerpunkt auf den besonders schwierigen Aspekten. Die im klickTel-Projekt angewendeten Lösungsstrategien sollen im Folgenden präsentiert werden. Vorbereitend wird dazu zunächst das Eingabeformat der Synthese beschrieben.

6.3.1.1. Eingabeformat

Bei der Eingabe, die von der Call-Center-Anwendung der Telefonauskunft erwartet wird, handelt es sich um ein XML-Dokument. Dieses Dokument wird mittels des in Abbildung 6.3.1 gezeigten Sende- und Empfangsmoduls ohne weitere Verarbeitung an die BOSS-Server-Applikation geschickt. Dort übernimmt die eingebettete CTS-Anwendung die Vorverarbeitung und Umsetzung in BOSS-Server-XML (vgl. Abschnitt 4.2.1). Das XML-Dokument enthält die in Abbildung 6.3.2 gezeigte Elementhierarchie.

Jeweils ein Anrufer bzw. eine Rückgabe des Auskunftssystems entspricht einem `<REQUEST>` und damit einem XML-File, da nur ein `<REQUEST>`-Tag pro Datei erlaubt ist. Ein `<REQUEST>` erhält genau ein `<ENTRY>`-Tag, das einem zurückgelieferten Eintrag entspricht und aus einer beliebigen Kombination von Eintragsfeldern (`<NAME>`, `<STRASSE>` etc.) in beliebiger Anordnung von `<SENTENCE>`-Strukturen bestehen kann. Alle Felder auf der untersten Ebene sind optional und auch nur dort existieren Attribute, nämlich `Text`, das den Inhalt des Namens- oder sonstigen Feldes enthält, und `PMode`, das wie in BOSS-XML steigenden und sinkenden Intontationsverlauf an der Phrasengrenze oder mediale Phrasenposition signalisieren kann. `<SENTENCE>`-Elemente und `PMode`-Attribute können flexibel zur Phrasierung der Äußerung eingesetzt werden. Dabei markiert ein schließendes `<SENTENCE>`-Tag immer eine finale Phrasengrenze, d.h. das vorausgehende `PMode`-Attribut muss auf `"."` gesetzt werden.

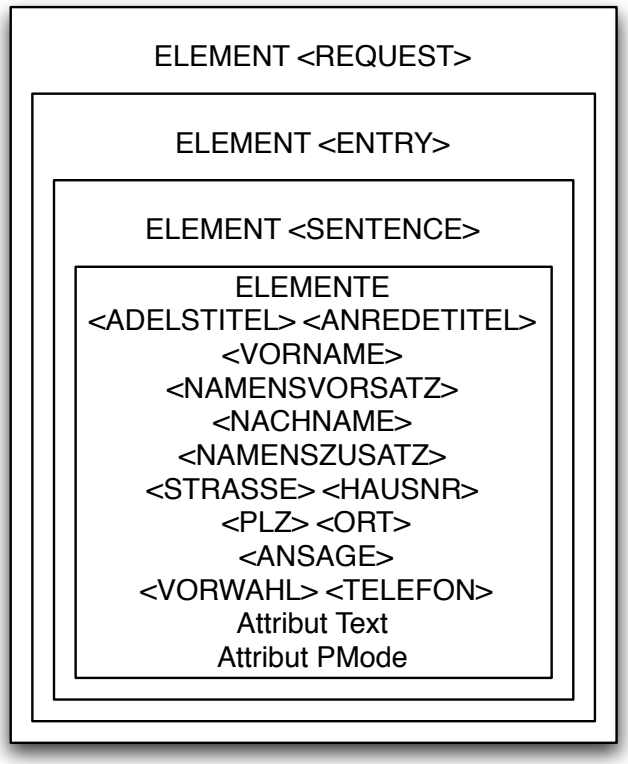

Abbildung 6.3.2.: Elementhierarchie einer klickTel-Eingabe für die Synthese.

```
1  <?xml version='1.0' encoding='ISO-8859-1' ?>
2  <REQUEST>
3          <ENTRY>
4                  <SENTENCE>
5                          <NACHNAME PMode="?" Text="Harald␣Schmidt"/>
6                          <STRASSE PMode="" Text="Theaterplatz"/>
7                          <HAUSNR PMode="?" Text="23"/>
8                          <PLZ PMode="" Text="25337"/>
9                          <ORT PMode="." Text="Unterkassel"/>
10                 </SENTENCE>
11                 <SENTENCE>
12                         <ANSAGE PMode="?" Text="Telefon"/>
13                         <VORWAHL PMode="?" Text="0555"/>
14                         <TELEFON PMode="." Text="987654"/>
15                 </SENTENCE>
16         </ENTRY>
17 </REQUEST>
```

Listing 6.1: Beispiel einer XML-Eingabe für die CTS-Komponente der klickTel-Synthese.

Satzintern dürfen für beliebige Felder mittels `PMode = "?"` interne, progrediente Phrasengrenzen markiert werden.

Listing 6.1 zeigt ein Beispiel für ein klickTel-Eingabedokument. Die dort verwendete Phrasierung entspricht derjenigen, die für die Anwendung festgeschrieben wurde und für die das Synthesekorpus optimiert war (vgl. Abschnitt 6.1.1).

6.3.1.2. Abkürzungsauflösung

In einem ersten Schritt wird für jedes Textfeld (im Unterschied zu den reinen Zahlenfeldern) eine Verarbeitung der Abkürzungen und Sonderzeichen vorgenommen. Dies geschieht durch Verarbeitung von regulären Ausdrücken, die beim Start der CTS-Komponente aus einer Datei namens `kt_regex.txt` ausgelesen werden. Die Externalisierung dieser Regeln ermöglichte die schnelle Anpassung und Erweiterung durch den Entwickler und die Auftraggeber. In der Datei sind jeweils drei Spalten pro Zeile enthalten, die den zu erkennenden String, dessen Substitution und die Kürzel für die Felder enthält, auf die der Ausdruck Anwendung finden soll, so dass eine von der Eintragsart abhängige Verarbeitung möglich ist:

```
Hr\. Herr_ ar,nn
```

Durch den gezeigten Ausdruck wird die Abkürzung `Hr.` für die Felder Anrede und Nachname (ar,nn) in `Herr_` umgesetzt. Hierbei steht der Unterstrich für ein Leerzeichen. Alle für einen Feldtyp definierten Ausdrücke werden in der gegebenen Reihenfolge abgearbeitet, daher ist darauf zu achten, dass die mächtigsten Ausdrücke möglichst weit unten in der Liste definiert werden. Weitere Beispiele für Nachnamen und Namenszusätze (nz) wären:

```
(\s|_|^)+Landkr\. _Landkreis_ nn,nz
(\s|_|^)+Lohnst\. _Lohnsteuer_ nn,nz
(\s|_|^)+Orthop\.  _[Orthop]_Punkt_ nn,nz
(\s|_|^)+Rheinl\. _Rheinland_ nn,nz
(\s|_|^)+Serv\.KG(\s|_|$)+ _Service_KG_ nn,nz
(\s|_|^)+St\.Anna(\s|_|$)+ _Sankt_Anna_ nn,nz
```

Die nächste Stufe der Textvorverarbeitung überprüft, ob und wie eine weitere Transformation vorgenommen werden muss, wenn durch Großschrift oder Punkt am Wortende eine Abkürzung signalisiert wird. Abbildung 6.3.3 zeigt die Vorgehensweise schematisch.

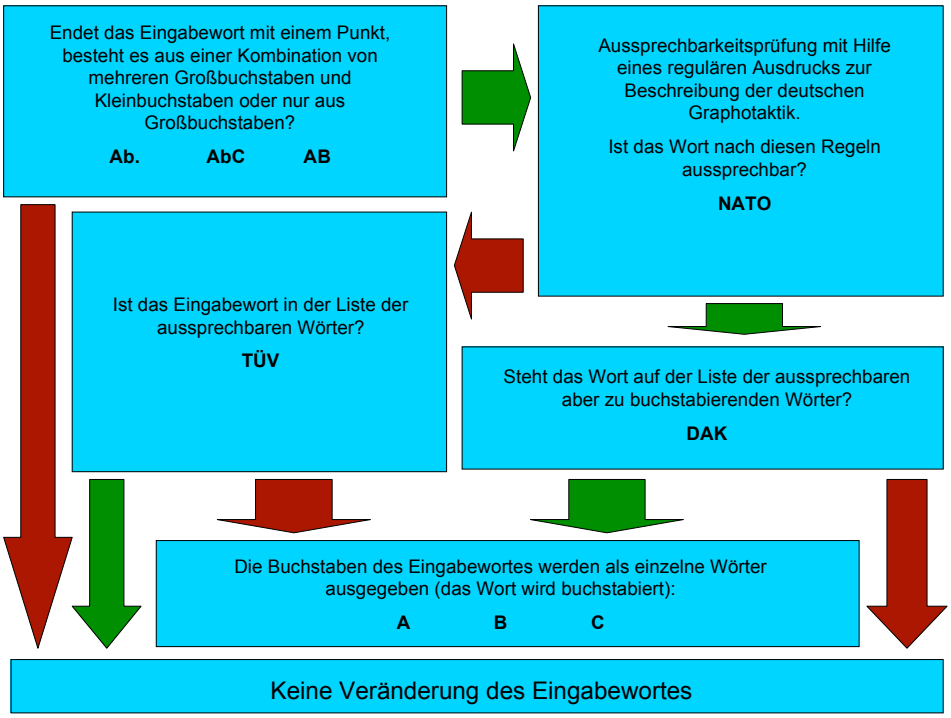

Abbildung 6.3.3.: Schema der Abkürzungsauflösung für die klickTel-Synthese.

Die Liste der aussprechbaren Namen entspricht dabei den orthographischen Einträgen des Lexikons ohne Transkriptionen, da die darin enthaltenen Wörter auf jeden Fall transkribiert werden können. Sie wird automatisch mit dem Programm zur Erstellung des Aussprachewörterbuchs erzeugt. Die Liste der Wörter, die möglicherweise aussprechbar sind aber buchstabiert werden sollen, wurde dagegen von Hand erstellt. Sie enthält auch Abkürzungen, die nach dem regulären Ausdruck zur Prüfung der Aussprechbarkeit nicht abgedeckt sind, aber unbedingt buchstabiert werden müssen, weil sie nicht eindeutig expandiert werden können (Beispiel: „Christ.“). Die Aussprechbarkeitsprüfung deckt nur ein als sicher angesehenes Minimum an Graphemkombinationen ab und wird durch folgenden Ausdruck ohne Beachtung der Groß- und Kleinschreibung realisiert:

```
^(([wrtzpsdfghjklyxcvbnm]){0,1}([euioaäöü]){1}([iuao]){0,1}\\
([nmrtpsdfkl]){0,2})+$
```

Für homographe Einträge, die in verschiedenen Eintragsarten (Feldern) verschieden ausgesprochen werden, wird die adäquate Transkription über die in der Lexikongenerierung automatisch erzeugten Listen schon in der CTS-Komponente ermittelt

und über das Attribut `WordTranscription` im BOSS-Server-XML an das Transkriptionsmodul im Server übergeben. Dies ist notwendig, weil die automatische Transkription selbst keine Möglichkeit zur Disambiguierung hat und das Gesamtlexikon daher keine orthographischen Duplikate enthalten darf.

6.3.1.3. Vorverarbeitung von Zahlen

Hausnummernfelder werden durch den folgenden Ausdruck in ihre Bestandteile zerlegt und in Phrasen aufgeteilt:

```
^ *([0-9]+) *([A-ZÜÖÄa-züöäß]*) *(([-/]) *(([0-9]*)\\
 *([A-ZÜÖÄa-züöäß]*) *)){0,1} *$
```

Die enthaltenen Zahlen werden mit Hilfe einer von Bröggelwirth geschriebenen Klasse zur Umwandlung von Ziffern in ausgeschriebene Zahlwörter behandelt. Die Algorithmen zur Zahlenkonvertierung werden auch für Ziffern in Textfeldern verwendet. Dazu zählen auch römische Ziffern, von denen zur Vermeidung von Fehlern nur eine kleine Untermenge umgesetzt werden soll, die durch folgenden Ausdruck beschrieben wird:

```
_+([XVI]{2,})(\\.|_)+
```

Alle Zahlen in Telefonnummernfeldern werden, mit Ausnahme von Sonderrufnummern wie „0800" nach den in Abschnitt 6.1.3 dargelegten Prinzipien in Phrasen gruppiert.

Auf statistische Verfahren, um beispielsweise zusätzlich Wortarten-Tagging oder syntaktisches Parsing durchzuführen, wurde nicht zurückgegriffen, weil passende annotierte Textkorpora fehlten und während der Projektlaufzeit unmöglich erstellt werden konnten. Der Erfolg solcher Maßnahmen im beschriebenen Einsatzgebiet ist allerdings mehr als fraglich, insbesondere in Bezug auf das Parsing, weil die syntaktischen Strukturen im Telefonbuch in ihrer Heterogenität typische *rare events* darstellen und damit das Problem der *data sparsity* oder Datenknappheit zum Tragen kommt.

6.3.2. Graphem-Phonem-Konvertierung

Wie gezeigt wurde, deckt das erstellte Aussprachewörterbuch zwar die am häufigsten im Telefonbuch vorkommenden Namen und Wörter ab, bietet jedoch keine Transkriptionen für die einzelnen seltenen Typen, die darin millionenfach vertreten sind.

Wie Vorleseautomaten, die für Texte verwendet werden, benötigte die klickTel-Synthese daher einen Mechanismus, der die orthographische Eingabe zur Laufzeit in eine Folge von phonetischen Symbolen umwandeln kann. Eine automatische Vortranskription, die offline erfolgt, wäre angesichts der sich stetig wandelnden Auskunftsdatenbank nicht ausreichend gewesen. Bevor das für BOSS implementierte und verwendete Verfahren beschrieben wird, soll zunächst eine kurze Einführung zu den Methoden der automatischen Transkription gegeben werden.

6.3.2.1. Verfahren der automatischen Transkription

Für die maschinelle Transkription oder Graphem-Phonem-Konvertierung existiert eine Vielzahl von Verfahren. Die ersten Ansätze waren regelbasiert (z. B. Stock, 1992, für das Deutsche; Elovitz et al., 1976, für das Englische). Die immer noch gebräuchlichen regelbasierten Verfahren verwenden durch phonetische Experten erstellte Sätze kontextsensitiver Abbildungsvorschriften von Graphemen auf Phone oder Phoneme[3]. Die Erstellung solcher Regelsätze erfordert allerdings einen großen Aufwand, und die Qualität der Abbildungsvorschriften ist schwer mess- und vergleichbar (s. Damper et al., 1999). Um die Anpassungszeit auf neue Sprachen zu minimieren, wurde nach automatischen Lernverfahren gesucht, die selbstständig die vielschichtigen Graphem-Phonem-Relationen erfassen können. Sejnowski und Rosenberg (1987) setzten als erste erfolgreich neuronale Netze zur Transkription englischer Wörter ein. Auch Hidden Markov-Modelle (HMM) werden teilweise zur Graphem-Phonem-Konvertierung eingesetzt (Riley, 1991). Eine weitere wichtige Gruppe bilden die sogenannten *Pronunciation-by-Analogy*-Verfahren (PbA). Im weiteren Sinne werden darunter gelegentlich alle datengetriebenen und datenbasierten Klassifikationsverfahren zur Graphem-Phonem-Konvertierung subsumiert (vgl. Bagshaw, 1998), sofern sie aus einem gegebenen Datenkorpus generalisierte Abbildungsregeln extrahieren (implizite Analogie) oder zur Laufzeit Analogien mit Hilfe unkomprimierter Aussprachelexika bilden (explizite Analogie, Damper et al., 1999). Die Terminologie ist hier allerdings sehr uneinheitlich. Zumindest die PbA im engeren Sinne und datengetriebene Verfahren wie das memory-based learning (Busser et al., 1999; Daelemans, 1999; Daelemans und van den Bosch, 1996a, 2001; Daelemans et al., 2001) weisen große Ähnlichkeiten auf, die eine solche Einteilung rechtfertigen würden. Beide Verfahrensweisen lassen sich als Kontextregeln umformulieren, die nach statistischen Kriterien sortiert sind.

[3]Die Bezeichnungen Graphem-Phonem-Konvertierung oder grapheme-to-phoneme conversion sind in der Literatur üblich und werden deshalb — etwas widerwillig — als Synonym für Graphem-Phon-Konvertierung übernommen, obwohl in der Regel phonetische und keine phonologischen Symbole erzeugt werden. Ob es sich bei den Ausgangsdaten um Grapheme oder Graphe handelt ist dagegen vom Standpunkt abhängig.

Bei der expliziten PbA (Marchand und Damper, 2000) werden alle Zerlegungen eines Eingabewortes sukzessive mit allen orthographischen Teilketten des Lexikons verglichen und alle so ermittelten Abbildungsmöglichkeiten in einen Graphen eingetragen. Jeder Knoten ist mit Kosten versehen, die nach der Wahrscheinlichkeit der jeweiligen Abbildung vergeben werden. Eine Entscheidungsfunktion wählt dann aus Pfaden (bzw. Abbildungen) minimaler Länge den kostengünstigsten (wahrscheinlichsten) aus.

Beim memory-based learning (MBL) wird in der Lernphase ein Fenster mit fester Breite (z.B. 9 Zeichen) über jedes Wort im Lexikon geschoben. Das Graphem in der Fenstermitte ist jeweils das betrachtete Zeichen, das einer Zielkategorie (einem Phonem) zugeordnet werden soll, die Zeichen links und rechts davon sind Kontextbedingungen für die Abbildung. Zusammen bilden sie den Merkmalsvektor. Aus der Menge der Merkmalsvektoren wird ein Baum aufgebaut, bei dem jede Ebene ein Kontextmerkmal (oder Kontextposition) darstellt und an dessen Blättern die Phoneme stehen. Die Ebenen sind nach einem statistischen Maß, dem Informationsgewinn (*information gain*, Daelemans und van den Bosch, 2001) hierarchisch angeordnet. Dabei erweist sich das betrachtete Graphem naturgemäß als wichtigste „Kontext"-Information, gefolgt von der ersten rechten Kontextposition, der ersten linken, der zweiten rechten usw. Bei der Klassifikation (Transkription) wird ein unbekanntes Eingabewort in mehrere Merkmalsvektoren zerlegt, für die durch Durchlaufen des Baumes je ein Phonem ausgegeben wird. Ist ein gegebener Kontext nicht im Baum gespeichert, wird die wahrscheinlichste Abbildung für den ähnlichsten Kontext gewählt. Die Konkatenation der Ausgabephoneme für jeden Merkmalsvektor ergibt die Transkription.

Sowohl für MBL als auch PbA ist ein *Alignment*-Schritt Vorbedingung, in dem für jedes Orthographie-Transkriptions-Paar im zugrunde liegenden Aussprachewörterbuch definiert wird, welches Graphem auf welches Phonem abgebildet wird (bei Überzahl der Grapheme können auch Abbildungen auf ,,Nullphoneme" vorkommen). Dies kann z.B. durch Zählung der Graphem-Phonem-Korrespondenzen in jedem möglichen alignment aller Wörter des Lexikons geschehen. Am Schluss wird für jedes Wort das alignment gewählt, das die größte Gesamtwahrscheinlichkeit aller Abbildungen aufweist.

Eine andere Herangehensweise zur automatischen Transkription ist die morphologische Zerlegung eines Eingabewortes. (Möbius, 1998) setzt hierfür *finite state transducers* (FST) ein, die Transkriptionen der Morpheme sind dabei vorher bekannt und müssen nur noch zusammengesetzt und mit Verkettungsregeln (postlexikalischen Regeln) bearbeitet werden. Durch die Berücksichtigung phonologischer Prozesse an Morphemgrenzen sind diese Verfahren den anderen Methoden prinzipiell vorzuziehen. PbA z. B. tendiert zur Selektion längster Teilketten bei der Analogie-

bildung, was bei Morphemgrenzen überschreitenden Ketten zu falschen Transkriptionen führen kann. Das Problem der Morphemzerlegungsverfahren in Bezug auf die hier betrachtete Anwendung in der Eigennamentranskription ist, dass eine annotierte und transkribierte Morphemliste bereits vorliegen muss. Für Eigennamen, deren morphologische Struktur nur zu geringen Teilen aus synchron noch produktiven Einheiten besteht, sind die bestehenden Morphemlisten nicht sinnvoll verwendbar.

6.3.2.2. Automatische Transkription von Eigennamen

Die automatische Transkription von Eigennamen stellt eine besondere Herausforderung dar. Regeln und statistische Verfahren, die für den allgemeinen Einsatz erstellt bzw. trainiert wurden, sind in der Regel ungeeignet für die Verwendung bei Namen (Macchi und Spiegel, 1990; Vitale, 1991). Der Grund hierfür sind die für Namen unterschiedlichen und stärker irregulären Konventionen zur Umsetzung von Graphemen in Phoneme. Synthesesysteme schneiden bei der Umsetzung von Eigennamen daher häufig schlecht ab.

Mitte der neunziger Jahre wurde im Rahmen des EU-Projektes Onomastica (Onomastica, 1995; Schmidt et al., 1993) versucht, durch die Erstellung großer multilingualer Namenslexika Abhilfe zu schaffen. Für 11 europäische Sprachen wurden die jeweils häufigsten 20-50.000 Namenseinträge aus den Telefonverzeichnissen der teilnehmenden Länder von Phonetikern manuell transkribiert. Diese wurden als Basis zur Erstellung maschineller Transkriptionsverfahren verwendet, mit denen die restlichen Einträge in Lautschrift umgesetzt wurden. Insgesamt wurden 8,5 Millionen Namenseinträge bearbeitet, der weitaus größere Teil davon automatisch. Die dazu eingesetzten Verfahren waren dabei von Projektpartner zu Projektpartner verschieden und deckten nahezu das gesamte Spektrum der Herangehensweisen ab. Ebenso variabel ist die Qualität der maschinell erzeugten Transkriptionen. Für das Schwedische (Gustafson, 1996) und das Deutsche (Mengel, 1994) wurde als Transkriptionsverfahren u. a. morphologische Zerlegung auf Basis der *two-level morphology*, TWOL (Haapalainen und Majorin, 1995; Karlsson, 1990; Koskenniemi, 1983), angewendet. Dabei wurde das schwedische System um manuell gewonnene Namensmorpheme erweitert, während für das deutsche System keine Anpassung an die Besonderheiten der Morphologie von Eigennamen vorgenommen wurde. Während die Wortfehlerrate der Transkriptionen für das Schwedische bei 5-7 % lag (Nachnamen), ergaben sich für das Deutsche Fehlerraten zwischen 12 % (Straßennamen) und 88,1 % (Vornamen). Die Wortfehlerrate des schwedischen Ansatzes gehört zu den besten, die im Rahmen von Onomastica erzielt wurden. Es zeigt sich also, dass mit einer entsprechend angepassten morphologischen Zerlegung auch für Eigennamen gute Resultate erzielt werden können. Es fehlt aber bisher ein Ansatz, der eine sprachenunabhängige

Gewinnung von Namensmorphemen ermöglicht und die aufwändige Erstellung von einzelsprachlichen Morphemlisten erleichtert. Im Rahmen des Onomastica-Projektes wurde dies leider nicht angestrebt, für einen morphologisch basierten Transkriptionsansatz stehen nur die manuell gewonnenen schwedischen Morpheme zur Verfügung.

Auch in der Dissertation von Belhoula (1996) wird ein regelbasiertes System zur Transkription von Eigennamen aus dem Münchener Telefonverzeichnis um morphologische Komponenten erweitert. Die Gewinnung der verwendeten Namensmorpheme ist nicht dokumentiert, so dass von einer manuellen Erstellung ausgegangen werden muss. Eine erfolgreiche Anwendung von Belhoulas Regeln auf Daten aus dem gesamten Bundesgebiet würde mit Sicherheit eine Erweiterung der Morphemliste erfordern - auch hier wäre deshalb ein automatisches Verfahren wünschenswert. Ein erster Ansatz für eine halbautomatische morphologische Zerlegung in der Anwendung auf deutsche Eigennamen wurde von Jannedy und Möbius (1997) präsentiert. Dazu wurden Teilwortketten von 3 bis 7 Zeichen aus Ortsnamen z. T. automatisch nach Häufigkeit und manuell nach muttersprachlicher Intuition selektiert und in dem von Möbius (s. o.) entwickelten FST-basierten Transkriptionsverfahren eingesetzt. Die Ergebnisse sind mit 11-13% Wortfehlerrate vielversprechend, bleiben aber noch hinter den mit manuell erstellten Morphemlisten erzielten zurück: Belhoula erreicht eine minimale Wortfehlerrate von 4,3% für Ortsnamen, 8,9% für Nachnamen, 12% für Vornamen und 9,7% für Straßennamen, Gustafson für schwedische Eigennamen 5-7%.

6.3.2.3. Auswahl eines Verfahrens

Wie in Abschnitt 4.2.3 beschrieben, verfügte BOSS bereits über eine Methode zur morphologischen Zerlegung für unbekannte Eingabewörter, die allerdings nur eine Liste von Morphemen deutscher Inhalts- und Funktionswörter verwendete. Da zu Beginn des Projektes noch keine Namenstranskriptionen vorlagen, konnten auch keine lautlich annotierten Morphembestandteile von Namen segmentiert und dieser Liste hinzugefügt werden. Zum anderen liefert die Morphemkomponente des BOSS-Transkriptionsmoduls nur eine Ausgabe, wenn alle Bestandteile des Wortes in der Liste verzeichnet sind. Für den Fall, dass ein im Aussprachewörterbuch nicht aufgeführtes Wort auch von der Morphemdekomposition nur teilweise oder gar nicht zerlegt werden konnte, existierte also keine Ausweichstrategie und das Wort konnte in der Konsequenz nicht synthetisiert werden. Außerdem kann die Morphemkomponente keine Vorhersagen zur Position des lexikalischen Akzents in einem transkribierten Wort machen. Es musste also zunächst ein allgemeineres Verfahren her, das auf graphemischer Basis aus den bereits vorhandenen Transkriptionsdaten, insbesondere BOMP, lernen konnte, die segmentale, silbische und prosodische Struktur

eines Eingabewortes zu ermitteln. Die Wahl fiel auf das oben geschilderte MBL-Verfahren, wie es von Daelemans und van den Bosch (1996b) beschrieben wird, mit den Erweiterungen um Silbengrenzen- und Akzentprädiktion nach Busser (1998). Die Umsetzung dieser Verfahren in eine Implementierung für BOSS wird im folgenden Abschnitt beschrieben.

6.3.2.4. Graphem-Phonem-Konvertierung für BOSS

Die Implementierung der Graphem-Phonem-Konvertierung in BOSS besteht aus einem Programm zur Erstellung der Entscheidungsbäume (`traing2p`) und einer Laufzeitkomponente zur Klassifikation und Transkription unbekannter Eingabewörter in Form der Klasse `BOSS_g2p` und ihrer sprachspezifischen Ableitungen.

Der Prozess der Erstellung eines Baumes durch `traing2p` beginnt mit der Zerlegung der orthographischen und phonetischen Eingabe in Buchstaben und Segmente. Alle Wörter werden zunächst in Kleinschreibung konvertiert und dann in einzelne Zeichen separiert. Da die Phone durch mehrere Zeichen repräsentiert werden können, wird für die Aufteilung auf eine Konfigurationsdatei `g2p_symbols.txt` zurückgegriffen, die alle Lautsymbole listet. Im nächsten Schritt folgt das Alignment: Für jedes Eingabepaar „orthographischer Text“ / „phonetische Transkription“ muss festgelegt werden, welcher Buchstabe auf welchen Laut abgebildet wird. Dafür werden zunächst für jedes Paar alle möglichen Abbildungen erzeugt. Ist der orthographische Text genauso lang wie die Transkription, gibt es nur eine Ausrichtung. Ansonsten werden „Nullphone” eingefügt, um verschiedene Abbildungen zu erzeugen:

```
S c h l a n g e
S l a N @ 0 0 0
S 0 l a N @ 0 0
S 0 0 l a N @ 0
. . .
```

Aus allen erzeugten Abbildungen aller Wörter wird eine Abbildungsmatrix mit den gewichteten Häufigkeiten der Graphem-Phonem-Beziehungen erstellt. Je größer der Positionsunterschied zwischen Buchstabe und Laut (ohne Alignment), desto schwächer die Gewichtung:

G/P	S	s	t
s	340	920	17
t	12	24	800

Nun wird mit Hilfe der Matrix für jedes Wort das plausibelste Alignment errechnet, indem die Werte für jede Graphem-Phonem-Abbildung jedes Alignments aufaddiert werden:

```
S c h l a n g e
S l a N @ 0 0 0  340+11+14+17+24+56+64+48 = 574
. . .
S 0 0 l a N 0 @  340+63+67+930+1100+530+64+678 = 3772
```

Das nach dieser Berechnung beste Alignment wird durch ein Fenster der ungeraden Länge n geschoben, hier gezeigt für den BOSS-Standardwert $n = 9$, der nach der Empfehlung von Busser (1998) bestimmt wurde. Jede Position im Fenster repräsentiert ein Merkmal, nämlich das abzubildende Graphem und seine $abs(\frac{n}{2})$ linken und rechten Kontextgrapheme. Die Ausgangsposition ist der Beginn des Graphemstrings in der Mitte des Fensters, die den abzubildenden Buchstaben repräsentiert. Dieser Position wird die nach dem Alignment ermittelte Abbildung vom Anfangsbuchstaben auf das Phon zugewiesen. Anschließend wandert der nächste Buchstabe in die Fenstermitte und wird auf den ihm zugeordneten Laut abgebildet. Dieser Vorgang wird wiederholt bis der letzte Buchstabe des Wortes im Zentrum steht. Die daraus entstehenden Merkmalsvektoren enthalten damit die Zuordnung eines Graphems in einem bestimmten Kontext zu einem Lautsymbol. Positionen, die über die Wortgrenze hinausgehen, werden dabei durch das Zeichen # belegt:

```
# # # # s c h l a   S
# # # s c h l a n   0
# # s c h l a n g   0
# s c h l a n g e   l
s c h l a n g e #   a
c h l a n g e # #   N
h l a n g e # # #   @
```

Um Speicherplatz zu sparen und den Zugriff zu beschleunigen, werden die Merkmalsvektoren in eine Baumstruktur umgewandelt, an deren Blättern die Zielphone stehen. Die Anordnung der Merkmale in der Baumhierarchie wird nach dem Informationszuwachs, *information gain*, IG, festgelegt, einem Maß für den Unterschied in der Entropie mit und ohne Merkmal. Sie ist für alle Merkmalsvektoren gleich, so dass eine Ebene des Baumes immer für ein Merkmal, z. B. den unmittelbaren rechten Kontext, steht. Die Berechnung des IG wird in Gleichung 6.3.1 nach Daelemans et al. (2000) wiedergegeben:

$$w_i = H(C) - \sum_{v \in V_i} P(v) \times H(C|v) \tag{6.3.1}$$

C ist hier die Menge der Klassen (Phone) und V_i repräsentiert die Menge der Werte (Grapheme), die ein Merkmal i (bspw. zweiter linker Kontext) annehmen kann. Die Entropie $H(C)$ ist in Gleichung 6.3.2 definiert.

$$H(C) = -\sum_{c \in C} P(c) \log_2 P(c) \tag{6.3.2}$$

Nach dem IG-Maß ist für das Deutsche der abgebildete Buchstabe — nicht überraschend — das wichtigste Merkmal. Darauf folgen abwechselnd direkte und indirekte rechte und linke Graphemkontexte in größer werdenden Abständen. Daelemans und van den Bosch (1996b) berichten, dass dieses Verhalten bei der Baumerstellung für alle von ihnen getesteten Korpora zutrifft.

Alle zur Disambiguierung einer Abbildung nicht benötigten Merkmals-Teilbäume werden entfernt. Dies entspricht nicht einem Pruning, da keine für die Entscheidung potenziell relevanten Daten entfernt werden. Sind mehrere widersprüchliche Merkmalsvektoren vorhanden, wird nach Auftretenshäufigkeit entschieden, welches Phon ein Blatt des Teilbaumes wird. Bei Gleichstand entscheidet der Zufall.

Die mit dem geschilderten Verfahren erzeugten Graphem-Phonem-Abbildungen enthalten keine Informationen über den Wortakzent und die Silbenstruktur, da sich das in BOSS verwendete Verfahren auf die empirischen Ergebnisse von Busser (1998) stützt, die einer konsekutiven Prädiktion dieser Parameter und der segmentalen Struktur eine bessere Performanz bescheinigt, als einer gleichzeitig stattfindenden. Das bedeutet, dass in BOSS separate Bäume für die Prädiktion von Phonen, Silbengrenzen und Akzenten trainiert werden, die jeweils die bereits als bekannt vorausgesetzten Parameter der vorherigen Stufe mit einbeziehen. Die Silbengrenzenzuordnungen werden also auf Basis der phonetischen Struktur ermittelt, während die Akzentzuweisung die silbifizierten Transkriptionen als Trainingsgrundlage verwendet. Der Ablauf des Trainings ist derselbe wie für die segmentale Zuordnung, allerdings wird kein Alignment benötigt. Die Merkmalsvektoren für die Silbifizierung werden so konstruiert, dass dem Phon in der Mitte des Fenster entweder ein Wert 1 oder 0 zugewiesen wird, je nachdem ob ihm eine Silbengrenze vorausgeht oder nicht.

```
# # S l a N @ #    0
# S l a N @ # #    1
```

Bei den Akzentvektoren sind es die Werte 0, 1, oder 2 für nicht vorhandenen, primären oder sekundären Akzent.

```
# # S l a . N @    1
# S l a . N @ #    0
```

Die Klassifikation (Transkription) mit dem Entscheidungsbaum durch `BOSS_g2p` zur Laufzeit der Synthese verläuft folgendermaßen. Analog zum Verfahren beim Lernen wird ein Eingabestring in Merkmalsvektoren (einen Vektor pro Buchstabe) zerlegt. Für jeden Vektor wird ein Pfad durch den Baum gesucht, der den Merkmalen entspricht. Das gefundene Blatt enthält das Zielphon. Die zusammengesetzten Zielphone bilden die Transkription:

```
# # # # e n d e #  ?E
# # # e n d e # #  n
# # e n d e # # #  d
# e n d e # # # #  @
```

Nach Ermittlung der segmentalen Struktur werden wiederum Merkmalsvektoren für den phonetischen String erstellt und der Baum für die Silbenprädiktion durchsucht. Das dort gefundene Ergebnis wird dann ebenfalls zerlegt und in einem letzten Prädiktionsschritt mit Wortakzentsymbolen versehen.

Da der Algorithmus nicht garantieren kann, dass die Phonotaktik der Zielsprache in den erzeugten Transkriptionen eingehalten wird, und auch nicht dafür Sorge trägt, dass die Silbengrenzen und Akzente an erlaubten Positionen eingefügt werden, wurde für jede Stufe der Prädiktion eine Nachverarbeitung programmiert, die z.B. unplausible Geminaten entfernt, doppelte, aufeinanderfolgende Silbengrenzsymbole eliminiert und Akzente ggf. vor den Silbenkern verschiebt. Da nur dieser Teil von `BOSS_g2p` sprachenspezifisch ist, wurden die Nachverarbeitungsfunktionen dort nur virtuell integriert und eine abgeleitete Klasse `BOSS_g2p_DE` geschaffen, die alle Eigenschaften der Oberklasse erbt und diese Funktionen für das Deutsche bzw. boss-sampa_de implementiert. Damit folgt der Aufbau den in Abschnitt 5.2.4 gegebenen Empfehlungen für die Aufteilung von Modulen. Da die deutsche G2P-Klasse wiederum als der letzte von drei Transkriptionsmechanismen in die Klasse `BOSS_Transcription_DE` eingebunden ist, benötigt sie keinerlei anwendungs- bzw. schnittstellenspezifischen Code zur XML-Kommunikation und kann daher auch anderweitig eingesetzt werden. Daher konnte dieses Klasse auch in ein eigenständiges Transkriptionsprogramm `boss_g2p_de` eingebunden werden, das unabhängig von BOSS funktioniert. Die Einbindung der graphembasierten Transkription wertete

auch die Morphemkomponente auf, da die Schnittstellen der Klasse auch erlauben, für einen Eingabestring nur bestimmte Stufen der Prädiktion zu verwenden. In `BOSS_Transcription_DE` wird die G2P-Klasse daher auch für die Akzentzuweisung der durch Morphemzerlegung entstandenen Transkriptionen eingesetzt.

Das Programm `train_g2p` und die Klasse `BOSS_g2p` verwendeten zunächst die MBL-Implementierung Tilburg Memory Based Learner (TiMBL) von Daelemans et al. (2000) zur Erstellung der Bäume aus den Merkmalsvektoren bzw. das Durchlaufen der Bäume. Diese wurde später wegen einer Inkompatibilität der Lizenz mit der für BOSS verwendeten erweiterten *GNU Public License* durch eine Implementierung von Groß ersetzt (`libphimbl`), die mit den in TiMBL erstellten Bäumen kompatibel ist und diese Bäume auch erzeugen kann.

Ursprünglich war angedacht, das G2P-Modul auch für die Morphemtranskription zu verwenden. Die Erstellung einer Morphemliste von Namen war im Zeitrahmen des Projektes allerdings nicht mehr möglich. Ein behelfsmäßiger Import des kompletten Lexikons im angepassten Format als Morphemliste in die bereits existierende Morphemkomponente wurde aufgrund der stattlichen Größe des Lexikons nicht durchgeführt. So bleibt das dargestellte Modul der einzige Fallback für die Transkription von Namen. Zwar stand auch für eine umfassende Evaluation der Transkription keine Zeit mehr zur Verfügung, nachträglich wurde jedoch ein Test zur Bestimmung der Wortfehlerrate mit einem Holdout von 690 zufällig ausgewählten verschiedenen Namen aus der Datenbank durchgeführt. Die Zahl ergibt sich aus der Auswahl von 250 Namen je Eintragsart und anschließender Entfernung von Homographen und Abkürzungen. Die in Tabelle 6.3.1 dargestellten Ergebnisse beziehen sich auf einen Vergleich ohne Berücksichtigung der Akzent- und Silbenstruktur. Da die Abweichung der Transkriptionen auf Phonebene nicht analysiert wurde, kann die Schwere der Wortfehler nicht abgeschätzt werden.

	Vornamen	Namen	Straßen	Orte
n	195	171	126	238
korrekt	141	123	112	194
falsch	54	48	14	44
Wortfehlerrate	27,69	28,07	11,11	18,49

Tabelle 6.3.1.: Wortfehlerraten der Graphem-Phonem-Konvertierung, aufgeteilt nach Eintragsarten.

Die ermittelten Werte sind mit einer Wortfehlerrate von 21,34% für die Transkription von Namen keinesfalls als katastrophal zu bezeichnen. Die in anderen Studien für Morphemzerlegungen erreichten Ergebnisse suggerieren jedoch, dass es richtig

war, eine solche Methode auch für die Telefonauskunftssynthese anzustreben. Unklar bleibt jedoch, ob mit denselben Verfahren diese Werte auch auf den klickTel-Daten zu reproduzieren wären.

6.3.3. Vorauswahl von Einheiten

Durch die Einführung der konfigurierbaren Vorauswahl konnte diese leicht an die Anwendung in der Telefonauskunftssynthese angepasst werden (vgl. Abschnitt). Für die Synthese der Adresseinträge schien es angebracht, die segmentale Verständlichkeit gegenüber der prosodischen Adäquatheit höher zu bewerten. Das bedeutete für die Vorauswahl, dass das Konzept des *longest matching* auch auf Kosten der prosodischen und auch kontextuellen Eigenschaften durchgesetzt wurde. Für jede Ebene der Auswahl wurden daher möglichst viele konsekutive Abfragen mit einer absteigenden Zahl von Kriterien definiert, bis der Rückgriff auf die nächstkleinere Ebene unausweichlich wurde, also nicht einmal das Kriterium der lautlichen Identität erfüllt wurde. Auf der Wortebene sind dies in der klickTel-Anwendung z. B. 38 verschiedene Kriteriengruppen, von denen hier nur ein kleiner Ausschnitt vom Anfang und Ende der Definition präsentiert wird (vgl. Abschnitte 4.1.1, 4.1.2 und 5.3.1 für eine Erläuterung der gezeigten Attribute sowie Abschnitt 6.3.3 für eine Definition der Syntax der Vorauswahl-Konfigurationsdateien):

```
TKey    TKey
CLeft   CLeft
CRight  CRight
PInt    PInt
Align   Manual

TKey    TKey
CCLeft  CCLeft
CRight  CRight
PInt    PInt
Align   Manual

TKey    TKey
CLeft   CLeft
CCRight CCRight
PInt    PInt
Align   Manual
 ...     ...
```

```
TKey    TKey
CCLeft  CCLeft
Align   Manual

TKey    TKey
PInt    PInt
Align   Manual

TKey    TKey
Align   Manual
```

Mit dieser Konfiguration gleicht die Pre-Selection von BOSS dem Clustering-Verfahren (s. Abschnitt 2.5.1), ohne dass eine akustische Beziehung zwischen den Kandidaten hergestellt wird. Die Baumdarstellung ist in der Abfrage effizienter als die hierarchische Abfrage von BOSS. Ein möglicher Pluspunkt der Hierarchie ist allerdings, dass für jede Eingabe dieselbe Gruppierung und Folge von Kriteriengruppen angewendet wird, während in den Bäumen nur dort Cluster erstellt werden, wo nach der akustischen Distanzmetrik ein Unterschied durch eine bestimmte Ausprägung eines Kriteriums festgestellt wird. Wie sich dieser Unterschied auf die Qualität der Auswahl auswirkt, wurde allerdings bisher nicht erforscht.

6.3.4. Kostenfunktionen

Die klickTel-Synthese verwendete die in Abschnitt 5.3.3 beschriebenen Erweiterungen der Kostenfunktionen. Zusätzlich wurde ein spezifischer Term zur Erhöhung der Verständlichkeit eingeführt.

Nach den in Kapitel 3 dargelegten Argumenten ist die enge phonetische Beschreibung des Sprachsignals ungeeignet für die Auswahl in der konkatenativen Synthese und wird auch in der Vorauswahl der klickTel-Version von BOSS nicht verwendet. Dennoch wurden im Verlauf der manuellen Korrektur der Segmentierung Abweichungen zusätzlich annotiert. Um diese Informationen in der Auswahl verwerten zu können, wurde eine Kostenfunktion erstellt, die im Falle ansonsten gleicher Eignung zweier Einheiten eine Entscheidung zugunsten des weniger stark reduzierten Segmentes ermöglicht. Diese Bewertung wurde nur auf der Phon-Ebene durchgeführt. Sie ist stark an die sprecherspezifischen Gegebenheiten im klickTel-Korpus ausgerichtet und erhebt keinen Anspruch auf generelle Verwendbarkeit oder Vollständigkeit im Sinne einer Modellierung von Reduktionsphänomenen im Deutschen. Formel 6.3.3 fasst die Kriterien dieses Kostenterms zusammen. Die Symbole _n und _\ stehen für Diakritika und bedeuten ,,nasaliert" bzw. ,,mit deutlicher Friktion realisiert". c

bezeichnet die resultierenden Kosten, u steht für eine betrachtete Kandidateneinheit und t für deren Zielvorgabe, das *target*.

$$c = \begin{cases} 500, & \text{wenn u} \neq \text{t} \\ 200, & \text{wenn u} = \text{"t\_n"} \bigwedge u+1 \neq \text{"@n"} \\ 100, & \text{wenn u} = \text{"t\_n"} \bigwedge u+1 = \text{"@n"} \\ 100, & \text{wenn u} = \text{"t\_\textbackslash"und linker und rechter Kontext Sibilant} \\ 200, & \text{wenn u} = \text{"t\_\textbackslash"und kein oder nur linker oder rechter Kontext Sibilant} \\ 200, & \text{wenn u entstimmt und kein stimmhafter Plosiv oder "z"} \\ 200, & \text{wenn u abweichend stimmhaft realisiert} \\ 100, & \text{wenn u nasaliert und linker und rechter Kontext nasal} \\ 200, & \text{wenn u nasaliert und kein oder nur linker oder rechter Kontext nasal} \\ 0, & \textit{sonst} \end{cases} \tag{6.3.3}$$

6.4. Evaluation der Synthese

Der Zeitplan des Auskunftsprojektes sah keine Evaluation der Synthesekomponenten vor und so konnten die Beiträge der einzelnen Entwicklungen zur Qualität nur informell abgeschätzt werden. Nichtsdestotrotz sollte gegen Ende der Laufzeit eine Evaluation des Gesamtsystems gestartet werden. Ziel der Evaluation sollte sein, die Einheit aus System und Korpus auf ihre Eignung für die Anwendung der Namenssynthese zu überprüfen und nicht eine optimale Einstellung der Syntheseparameter zu finden (vgl. auch Abschnitt 2.5 zu den Schwierigkeiten dieser Maßnahme). Als Ausgangspunkt sollte daher eine während der Entwicklung als geeignet befundene Kombination von Kostentermen und Gewichten festgelegt und während der Evaluation nicht variiert werden. Trotz dieser Festlegung war die Ermittlung der Eignung nicht trivial, denn ein weiteres großes Problem der Evaluation von Unit-Selection-Systemen ist, dass die für die konkatenativen Systeme mit einer 1:1-Relation von Einheitentypen und -instanzen entwickelten Techniken zur diagnostischen Evaluation einzelner Lautkombinationen nicht mehr greifen: Die Analyse einer bestimmten Instanz lässt in der Unit Selection nicht mehr auf die Gesamtqualität schließen, da für denselben Laut möglicherweise viele hundert Alternativen zur Verfügung stehen. Da das Problem der Diagnostik von Einzellauten nicht gelöst werden konnte, wurde versucht, durch Fokus auf die in der Telefondatenbank häufigsten Triphonkontexte (soweit durch Transkriptionen verfügbar) wenigstens eine Gesamtperformanz für die

in der ,,Domäne" wichtigsten Ereignisse zu messen. Dabei musste von der naiven Annahme ausgegangen werden, dass die häufigen Einträge auch die Häufigkeit der Anfragen an die Auskunft repräsentieren, weil zu letzterer keine Daten vorlagen. Eine Analyse der Performanz für *rare events* bewegt sich außerhalb der Machbarkeit einer subjektiven Evaluation, so dass hierzu keine Aussage getroffen werden konnte und sollte.

Die Evaluation wurde von (Hammerstingl, 2003) im Rahmen seiner Magisterarbeit durchgeführt. Er verwendete die zuvor ermittelten häufigsten Triphonkontexte als Namensbestandteile in synthetischen Stimuli, um verschiedene Konditionen der Generierung in Präferenz- und Verständlichkeitstest zu evaluieren. Um einerseits die Worterkennung nicht zum dominierenden Faktor der Verständlichkeit zu machen und andererseits zu vermeiden, dass das Gros der Einheiten aus seinen vollständigen Originalkontexten reproduziert wurde, wählte er nach Datenbanklage möglichst seltene Namen, die diese Triphonkontexte enthielten, für die Synthese aus. Eine Übersicht der Ergebnisse findet sich bei Hammerstingl und Breuer (2003, 2004). Hier sollen nur die wichtigsten Resultate des Verständlichkeitstest dargestellt werden.

Wie bereits geschildert vermittelte die Ausgabe der Synthese den Eindruck eher schneller Sprache, der auch nach dem Pruning in abgeschwächter Form noch vorhanden war. Daher sollte im Rahmen der Evaluation auch getestet werden, wie sich eine Dauermanipulation auf die Verständlichkeit auswirkt. Gemessen wurde die Phonemfehlerrate und die Namensfehlerrate für die in allen getesteten Stimuli enthaltenen Namen. Die Berechnung dieser Werte ist in den Formeln 6.4.1 und 6.4.2 nach Hammerstingl und Breuer (2004) dargestellt:

$$Ph_f = \frac{(\text{Einfügungen} + \text{Auslassungen} + \text{Substitutionen}) \cdot 100}{\text{Anzahl der gesamten Phoneme} \cdot \text{Anzahl der Versuchspersonen}} \tag{6.4.1}$$

$$Na_f = \frac{(\text{Namen mit Verständlichkeitsfehlern} \geq 1) \cdot 100\%}{\text{Anzahl aller Namen} \cdot \text{Anzahl der Versuchspersonen}} \tag{6.4.2}$$

Den sechs Versuchspersonen des Verständlichkeitstests wurden je 51 Stimuli und Triphonkontexte für jede der Eintragsarten ,,Vorname", ,,Name", ,,Straße" und ,,Ort" präsentiert, Diese waren so in einen Trägersatz eingebettet, dass die prosodische Struktur der Zielstruktur der Auskunftssynthese entsprach. Für die Verständlichkeitsmessung wurde eine offene Antwortform gewählt, bei der die Versuchspersonen den Namen sowohl aufschreiben als auch nachsprechen mussten. In Konfliktfällen wurde die Aufzeichnung mit der niedergeschriebenen Variante verglichen. Keine der Versuchspersonen war phonetisch geschult.

Die Ergebnisse der Test bescheinigen der Synthese eine Phonemfehlerrate von 1,817 und 2,259% mit und ohne Dauermanipulation. Die Namensfehlerrate liegt bei 10,866 bzw. 12,5%. Obwohl die Unterschiede in der Bewertung nicht signifikant waren, wurde entschieden, die Dauern der synthetischen Signale generell in Richtung der prädizierten Werte zu manipulieren. Da sich die in Abschnitt 5.2.6 beschriebenen `ConMan`-Module noch in der Entwicklung befanden, wurde dazu auf Praat zurückgegriffen. Bei der Manipulation wurde darauf geachtet, dass die Dauerwerte der Einheiten mindestens den Faktor 1,2 und maximal 1,4 annahmen, die Ausgaben also immer gelängt wurden.

Die Ergebnisse attestieren der Telefonauskunftssynthese auf BOSS-Basis eine für die Namendomäne gute Verständlichkeit unter den o. g. Bedingungen. Allerdings kann nur der Einsatz im Tagesgeschäft eines Teledienstes zeigen, wie es um die Gesamtqualität des Systems bestellt ist.

7. Zusammenfassung und Ausblick

,,Man merkt nie, was schon getan wurde;
man sieht immer nur das, was noch zu tun bleibt.“ *Marie Curie*

,,Multilinguale und multifunktionale Unit-Selection Sprachsynthese: Designprinzipien für Architektur und Sprachbausteine” lautet der Titel der vorliegenden Arbeit und enthält damit schon einen Überblick über die vertretenen Kernthemen, wie sie in den Kapiteln 3, 5 und 6 behandelt wurden. So wurde in Kapitel 3 ein Weg aufgezeigt, wie die problematische explizite Modellierung satzphonetischer Phänomene durch den Einsatz spezieller Multiphon-Sprachbausteine umgangen werden kann. Im weiteren Verlauf wurde die Weiterentwicklung des *Bonn Open Synthesis System*, BOSS, dokumentiert und Prinzipien zur Konzeption einer Architektur vorgestellt, die eine vereinfachte und parallele Anpassung für unterschiedliche Sprachen und Funktionen bzw. Anwendungen erlaubt (Kapitel 5). Gleichzeitig wurde die Implementierung dieser Eigenschaften im BOSS-System dargestellt. Welche Aspekte bei der konkreten Adaption an eine bestimmte Aufgabe zu beachten sind, wurde in Kapitel 6 anhand des Beispiels einer Telefonauskunftssynthese beschrieben. Ein Schwerpunkt lag dabei auf der Erstellung der linguistischen und akustischen Ressourcen. Zum anderen wurde aber auch die Problematik der Textvorverarbeitung und Transkription anhand der gewählten Lösungsstrategien diskutiert.

Im Rahmen dieser Darstellungen und auch in den Grundlagenkapiteln wurde dabei immer auch auf die Grenzen der Text-to-Speech-Sprachsynthese hingewiesen. Gerade dort wo lexikalische Einheiten aus verschiedenen Sprachen oder Regionen zusammentreffen, wie es bei der in Kapitel 6 beschriebenen Aufgabe der Fall ist, wird die Uneinheitlichkeit der orthographischen Repräsentation von Sprache deutlich. In diesen Fällen können Fortschritte nur durch geballtes Weltwissen in Form von umfassenden Aussprachewörterbüchern erzielt werden. Das Problem der Homographie wird aber auch dadurch nicht in den Griff bekommen zu sein. Für Wörter aus verschiedenen Sprachen mag eine Sprachenidentifizierung durch Graphemstatistiken helfen. Treten diese aber isoliert auf, so wie die Namen in der Auskunftsanwendung, wird auch damit keine Disambiguierung möglich, so dass die Methode für dieses Projekt nicht in Frage kam. Die Probleme der Orthographie-Laut-Beziehung betreffen

allerdings keinesfalls nur seltene Namen oder andere Instanzen der im Laufe der Arbeit häufig genannten *rare events*. Die phonetisch unterspezifizierte Repräsentation bedeutet auch, dass viele Informationen nicht im Text enthalten sind, sondern durch Regeln oder statistisch erlernte Beziehungen zwischen Text und Phonetik ergänzt werden. Dies limitiert die Qualität der Synthese bereits in zweierlei Weise, denn erstens ist die Modellierung fehlerbehaftet und zweitens erfasst sie erst gar nicht alle Parameter, die für die Sprachproduktion relevant sind. Hier setzt die Unit Selection an, die versucht, die Variabilität der Sprache nur implizit durch die Wiedergabe aus einer möglichst großen Auswahl natürlicher Bausteine abzubilden. Das Problem dieses Verfahrens liegt dabei eher in der mangelnden Kontrolle über die Ausprägung der Variation. Viel mehr aber noch beschränkt die Kombinatorik der variierenden Sprachparameter die Qualität, da kein Korpus alle Varianten abdecken kann, die zur Erzeugung wirklich natürlich klingender Sprache notwendig wären. Potenzial zur Verbesserung besteht daher hauptsächlich in der Vergrößerung der Datenbasen und der Erweiterung phonetischer Modelle.

Wegen der gezeigten Grenzen des konkatenativen Ansatzes ist es umso wichtiger, dass mit dem hier beschriebenen Ausbau von BOSS auch andere Syntheseverfahren modular eingebunden werden können, ohne die BOSS-Anwendung abändern zu müssen. Interessant für die Zukunft ist hier vor allem die artikulatorische Synthese, die näher an den physiologischen Prozessen der Sprachproduktion angesiedelt ist als die Unit Selection. Ein Einsatz dieses Verfahrens im Bereich der Vorleseautomaten scheint durch die jüngeren Entwicklungen jedenfalls in greifbare Nähre zu rücken. Aber auch zur phonetischen Erforschung der Artikulation können diese Verfahren in BOSS eingebaut werden. Es müsste dann ein alternatives Vorverarbeitungsmodul geschaffen werden, das die Brücke zwischen Symbol und Signal auf einer höheren Ebene schlägt. Anstatt Grapheme in Phoneme umzuwandeln, würde dieses phonologische Repräsentationen in artikulatorische Gesten umwandeln und die akustische Generierung der Synthese überlassen. Dann wird auch die in Kapitel 3 kritisierte Modellierung der Reduktion und Assimilation wieder plausibel, weil die symbolische Darstellung direkt in Gesten umgesetzt werden könnte, anstatt in eine relativ abstrakte, möglicherweise uneinheitliche und daher wenig aussagekräftige Beschreibung von diskreten Folgen akustischer Ereignisse.

Aber auch die Unit-Selection bietet noch Raum für Verbesserungen und bleibt auf absehbare Zeit das beste Verfahren für Text-to-Speech. Daher ist auch die Erweiterung des BOSS-Systems zur Multilingualität ein wichtiger Schritt, denn es ist zu erwarten ist, dass in Zukunft für eine Vielzahl von Sprachen, für die noch keine Sprachsyntheseanwendungen existieren, eine solche nach Unit-Selection-Prinzipien erstellt wird. Wenn nicht sogar das BOSS-System selbst, so können doch wenigstens die Anregungen zur Architektur eines multilingualen Systems dabei Starthilfe geben.

Die Optimierung der Einheitenauswahlsysteme ist auch deshalb noch nicht ausgereizt, weil bisher nicht großflächig erforscht wurde, welche Algorithmen in welcher Kombination die besten Ergebnisse garantieren. Einen Ansatz, in einem breiteren Rahmen nicht nur ganze Systeme, sondern auch Teilkomponenten zu testen, stellt die Evaluationskampagne des *European Center of Excellence for Speech Synthesis*, ECESS, dar. Durch die Auswahl der besten Verfahren und einer möglichst generischen Implementierung könnte die sprachenunabhängige Basis von BOSS, das in Kapitel 5 vorgestellte minimale multinlinguale TTS-System, so erweitert werden, dass auch mit reiner Anpassung der Modelldaten und Korpora ohne Modifikation der Komponenten eine gute Qualität für verschiedene Sprachen erreicht wird. Im Bereich der automatischen Transkription werden aufgrund der oben geschilderten Probleme vermutlich global keine großen Unterschiede in der Performanz verschiedener evaluierter Teil-Systeme feststellbar sein. Für das BOSS-System können jedoch noch Verbesserungen umgesetzt werden. Im Fall der deutschen Synthese ist durch Erweiterung und Korrektur der Trainingsdaten noch eine Qualitätssteigerung zu erwarten. Einen eigenen Ansatz des IfK, den besten Algorithmus für ein Teilproblem der Synthese zu finden, stellt die parallele Implementierung verschiedener Methoden zur Intonationsprädiktion in BOSS dar (Kapitel 5). Nach Abschluss der Entwicklung sollen diese einzeln und später vergleichend evaluiert werden. Ein weitere Aufgabe für die Zukunft wird sein, die konsequente Umsetzung der präsentierten Empfehlungen zur Modulstruktur auch in älteren BOSS-Bestandteilen voranzutreiben.

Als Test für die multilinguale Anwendbarkeit des erweiterten BOSS-Systems steht auch die mitentwickelte polnische Version zur Verfügung. Neben dem Ausbau der sprachspezifischen Anpassungen der Module dieses Systems wäre es auch interessant, die bereits in Ansätzen bestehende Anwendung des Multiphon-Konzepts auf die polnische Sprache zu evaluieren. Aber auch die in der deutschen Spezifikation enthaltenen Einheiten sollen einer genaueren Untersuchung unterzogen werden, insbesondere in Bezug auf die Auswirkung einzelner Typen. Aufbauend darauf wird der nächste Schritt der bereits in Kapitel 5 angekündigte Vergleich mit uniformen Einheitentypen verschiedener Größen sein. Dazu ist eine Erweiterung der BOSS-Auswahlalgorithmen nach den ebenfalls dort dargestellten Überlegungen nötig. Aber auch der Einbau von konkurrierenden Verfahren, wie dem in Kapitel 2 beschriebenen Clustering wäre zum Vergleich der Verfahren interessant, da im Rahmen der ECESS-Evaluation nicht die Interna der Unit-Selection-Algorithmen evaluiert werden sollen. Wenn wie oben dargelegt die Verbesserung der Synthesequalität hauptsächlich über die Vergrößerung der Sprachkorpora erfolgt, muss die verstärkte Beschäftigung mit den Auswahlalgorithmen ohnehin eine Priorität der Forschung sein. Wegen der damit verbundenen hohen Berechnungskomplexität sind trotz beständiger Leistungssteigerung der Rechner optimierte Verfahren zur Unit Selection notwendig.

Das BOSS-System in seiner bestehenden Form bietet bereits eine flexible Architektur für die Entwicklung verschiedenster Syntheseapplikationen. Mit der stetigen Bemühung um Portabilität und Universalität durch die Erweiterung mit alternativen Syntheseverfahren ist auch die Hoffnung verknüpft, dass BOSS weiteren Institutionen eine Plattform für Syntheseentwicklung und phonetische Forschung sein kann und sich daraus Kooperationen in unterschiedlichen Bereichen der Sprachproduktionsforschung und Maschine-Mensch-Kommunikation entwickeln werden.

Abbildungsverzeichnis

2.0.1. Stark vereinfachtes Schema eines Vorleseautomaten nach dem Prinzip der konkatenativen Synthese. 6
2.0.2. Synthese des Wortes „Strolch" [ʃtrɔlç] mit Hilfe verschiedener phonetischer Einheiten. (P) Beliebiger Signalparameter (hier gezeichnet: Pegel und Stimmhaftigkeit) – Hess (2002) 7
2.1.1. Zerlegung der Eingabe /hanagasakidashita/ in alle möglichen Lautfolgen mit anschließender Auswahl passender nicht-uniformer Einheiten aus dem Korpus – modifizierte Darstellung nach Sagisaka (1988). 9
2.1.2. Aufbau des ersten Systems nach dem Prinzip der Non-Uniform-Unit-Selection. Der geklammerte Text oben entspricht der Symbolverarbeitungskomponente, die Kästen links ersetzen die Verkettungs- und Syntheseanteile der traditionellen konkatenativen Synthese – modifiziert nach Sagisaka (1988). 10
2.1.3. Anwendung der Kostenterme des ν-Talk-Systems – modifizierte Darstellung nach Iwahashi et al. (1992). 13
2.2.1. Schematische Darstellung des Konzeptes der Einheiten- und Transitionskosten – nach Campbell und Black (1996). 15
2.2.2. Berechnung der prosodischen und phonetischen Unterschiede zwischen ausgewählten Kandidaten und ihrem jeweiligen Ursprungskontext in der Datenbank innerhalb der Transitionskostenfunktion. Abb. aufbauend auf Black und Campbell, 1995) 16
2.3.1. Graphendarstellung der Einheitenkandidaten (oben) und rekursive Berechnung der optimalen Einheitenfolge (unten) in BOSS/Verbmobil. 21
2.5.1. Hypothetischer Ausschnitt eines Entscheidungsbaumes zum Clustering von Syntheseeinheiten. 27
2.5.2. Metrisch-phonologische Darstellung eines Inventarsatzes als Grundlage für *phonological structure matching* – Taylor und Black (1999). . 29

3.2.1. [ja] in der Folge /n jan/ – US10020008 43
3.2.2. [t.jeː] in der Folge /t.jeːm/ – US10020649 43
3.2.3. [iː.jeː] in der Folge /iː.jeː.k/ – US10020175 44
3.2.4. [o̯ː] in der Folge /n hoː.t/ – a00026 45

3.2.5. [ɦ] in der Folge /n heːr‖/ – a00755 45
3.2.6. [h] in der Folge /s hiːr‖/ – a00163 46
3.2.7. [a͡ːˑ] in der Folge /ər ʔaːˑb/ – US10020004 46
3.2.8. [a͡ɪ] in der Folge /r ʔaɪm/ – US10020088 47
3.2.9. [ʔa͡ʊ] in der Folge /t ʔaʊf/ – US10020004 47
3.2.10. [lɛ] in der Folge /ɔʏ.lɛt/ – US10020010 48
3.2.11. [s.lɛ] in der Folge /s.lɛn – US10020344 49
3.2.12. [lʊ] in der Folge /plʊs/ – a00678 49
3.2.13. [lɛ] in der Folge /a.len/ – a00313 50
3.2.14. [ʁ] in der Folge /ə raɪts/ – US10020080 50
3.2.15. [ʁ] in der Folge /draɪ.s/ – US100020009 51
3.2.16. [ʁ̥] in der Folge /traːˑ.s/ – US10020009 51
3.2.17. [ʁ̞] in der Folge /fraʊ.ə/ – US10020022 52
3.2.18. [aːɐ] in der Folge /daːr b/ – US10020186 52
3.2.19. [aɐ] in der Folge /ʔaːr.b/ – US10020058 53
3.2.20. [aːʁ] in der Folge /faːrt/ – US10020004 53
3.2.21. [aʁ] in der Folge /mar.t/ – US10020004 54
3.2.22. [voː] in der Folge /f voːk/ – US10020005 54
3.2.23. [ʊoː] in der Folge /k voːl/ – US10020249 55
3.2.24. [ə.n] in der Folge /sə nɔʏ/ – US10020017 55
3.2.25. [ɛn] (nicht als solches annotiert) in der Folge /nən p/ – US10020030 . 56
56figure.72
3.2.27. [ən], ([ə] nur rudimentär vorhanden) in der Folge /gən f/ – US10020020 57
3.2.28. [n̩] in der Folge /dən.d/ – US10020001 57
3.2.29. Multiphoneinheiten aus Konsonant-Vokalkombinationen in phoxsy. . 58
3.2.30. Multiphone aus Schwa plus silbisch realisierbaren Nasalen und Liquiden — mit und ohne Anfangskonsonanten. 59

4.1.1. Beziehung zwischen der Repräsentation einer Signalannotation durch BLF und der XML-Kodierung. 69
4.1.2. Ablauf der Korpusaufbereitung in BOSS (Klabbers und Stöber, 2001). 70
4.1.3. XML-Kodierung der Annotationsdaten und Darstellung in einer relationalen Datenbank. 71
4.2.1. Architektur des Laufzeit-Systems von BOSS II. Hypothetische Client-Programme sind *kursiv* beschriftet. 72

5.1.1. Elementhierarchie einer BOSS-Konfigurationsdatei 83
5.1.2. Ablauf des Netzwerkprotokolls für den Austausch zwischen BOSS-Client und -Server. Neue Datenfelder sind durch einen blauen Rahmen gekennzeichnet. 86

5.1.3. Ausschnitt aus der deutschen XML-Schema-Definition eines BOSS-Server-Dokuments mit Fokus auf der Satzebene. Die neuen Attribute sind im Diagramm blau eingefärbt. 88
5.2.1. Unterschiede in der Klassenaufteilung zwischen der ersten Implementierung der Dauerprädiktion und der aktuellen Version. Pfeile mit unausgefüllten Spitzen führen zu den jeweiligen Oberklassen. Durchkreuzte Pfeile zeigen auf innerhalb der Klassen implementierte C-*structs*. 91
5.2.2. Funktionale Komponenten einer Syntheseanwendung und ihre Wechselwirkungen 94
5.2.3. UML-Diagramm eines prototypischen BOSS-Moduls mit abstrakter mengentheoretischer Darstellung der Code-Aufteilung. Optionale Elemente sind grau dargestellt. 98
5.2.4. UML-2-Sequenzdiagramm eines Synthesevorganges. 102
5.2.5. Einfaches Glättungsverfahren für Verkettungsstellen (Prudon, 2002). 106
5.2.6. Verkettung der Phone [ɛ] und [n] ohne (oben) und mit Glättung (unten). 107
5.2.7. UML-Diagramm der BOSS-Klassen für die Konkatenation und Manipulation der Sprachsignale. 108
5.3.1. Einfaches Beispiel für die Konfiguration der Einheitenvorauswahl. . . 111
5.3.2. Beispiel für eine gescheiterte Auswahl von Diphonen nach Halbphon-Vorauswahl. D ist die Menge aller Diphone der Lautfolge a, b. Dargestellt sind nur die jeweiligen Werte des `Stress`-Attributs. Die gesuchte Folge a, b ist mit den gewünschten `Stress`-Parametern als Diphon nicht in der Datenbank vorhanden. Die auf Halbphonen basierende Pre-Selection wählt pro Halbphon eine nach den Vorgaben optimale Einheit aus. Dadurch wird eine Auswahl eines Diphons der Folge unmöglich, was durch die gestrichelten roten Kanten im Unit-Selection-Graphen symbolisiert wird. 119
5.5.1. Architektur des Laufzeit-Systems von BOSS 3. 126

6.1.1. Screenshot des Programms zur Auswahl von Namen für die Erzeugung der Lesevorlagen. 133
6.1.2. Abdeckung der nach Eintragsarten sortierten Instanzen der Volleinträge in der Telefondatenbank durch die Worteinheiten des Korpus vor und nach dem Pruning (Alle/Verbleibende) in Prozent. Die Absolutwerte finden sich im Anhang in Tabelle E.2.1. 138

6.1.3. Abdeckung der nach Eintragsarten sortierten Typen der Volleinträge in der Telefondatenbank durch die Worteinheiten des Korpus vor und nach dem Pruning (Alle/Verbleibende) in Prozent. Die Absolutwerte finden sich im Anhang in Tabelle E.2.1. 139
6.2.1. Grafische Oberfläche des Programms zur Erstellung von Namenstranskriptionen. 145
6.2.2. Abdeckung der nach Eintragsarten sortierten Instanzen der Volleinträge im Telefonbuch durch das Aussprachelexikon und Gesamtabdeckung. 147
6.3.1. Architektur des Laufzeit-Systems der klickTel-Synthese. 148
6.3.2. Elementhierarchie einer klickTel-Eingabe für die Synthese. 150
6.3.3. Schema der Abkürzungsauflösung für die klickTel-Synthese. 152

Tabellenverzeichnis

3.3.1. Zur Resynthese verwendete Attribute und ihre Berücksichtigung in den verschiedenen Stufen der Einheitenauswahl. Die Zahlen stellen die Gewichtungsfaktoren der Einheitenkosten (unit cost weights, *ucw*) dar. PS=Pre-Selection, US=Unit-Selection 61

3.3.2. Stimuluspaare und zugehörige phoxsy-Einheiten:*Sätze und Einheiten, die inkonsistent beurteilt wurden*; phoxsy-Einheiten, die aus den gleichen Korpuseinheiten zusammengesetzt wurden wie die boss-sampa-Gegenstücke (auch bei inkonsistenter Beurteilung schwarz gesetzt); **phoxsy-Einheiten, die aus verschiedenen Einheiten zusammengesetzt wurden.** Nach Breuer und Abresch (2004). 63

3.3.3. Signifikanzwerte für verschiedene Untermengen der Ergebnisse (CON = konsistent, INC = inkonsistent, SUB = Probanden, STIM = Stimuli). Nach Breuer und Abresch (2004). 64

4.1.1. Wertebereiche der Attribute `PMode` und `PInt` zur Beschreibung von Phrasengrenzen und die für die deutschen Datenbanken genutzten Kombinationen. 68

5.2.1. Die für das Deutsche verwendeten Dauervorhersageparameter. Die erste Spalte gibt den *feature*-Namen im CART an, die zweite das korrespondierende Attribut in BOSS-XML. In der dritten Spalte wird die Semantik des Parameters beschrieben. 89

5.3.1. Attribute zur Speicherung der Klassenzuordnungen des segmentalen Kontextes. 110

5.3.2. Beziehung zwischen den Attributen und Attributwerten der `<PHONE>`- und `<HALFPHONE>`-Ebenen in den Repräsentationen für Server und Inventar. 115

6.1.1. Aufbau der Tabelle `haupt` in der Projektdatenbank mit Erläuterungen und Beispieleinträgen. 131

6.1.2. Quartile der Lautdauer (ms) für den Multiphon-Typ `hI`. 136

6.1.3. Anzahl der verschiedenen Einheitentypen vor und nach dem Pruning. 137

6.3.1. Wortfehlerraten der Graphem-Phonem-Konvertierung, aufgeteilt nach Eintragsarten. 162

A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen 202
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 203
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 204
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 205
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 206
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 207
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 208
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 209
A.2.1. Phoxsy-Einheiten mit ihren Kontextklassen *(Fortsetzung)* 210

B.1.1. Verarbeitung verschiedener Attributwerte in BOSS-Konfigurationsdateien durch die Klasse `BOSS::Config` (vgl. Abschnitt 5.1.1). 211

E.2.1. Abdeckung der Volltexteinträge in der Projektdatenbank durch die Worteinheiten im klickTel-Korpus, sortiert nach Eintragsarten und verschiedenen Phrasenpositionen (vor und nach dem Pruning) — vgl. Abschnitt 6.1.5. 239
E.3.1. Abdeckungsstatistiken für die Voll- und Teileinträge der klickTel-Projektdatenbank durch die Transkriptionen im Aussprachewörterbuch (vgl. Abschnitt 6.2.5). 239

Literaturverzeichnis

Abresch, J. (2007): *Englisches in gesprochenem Deutsch: Eine empirische Analyse der Aussprache und Beurteilung englischer Laute im Deutschen.* Dissertation, Universität Bonn.
URL `http://hss.ulb.uni-bonn.de/diss_online/phil_fak/2007/abresch_julia/abresch.htm`

Abresch, J.; S. Breuer (2004): "Unit-Selection-Sprachsynthese für die Telefonauskunft." In: *IKP-Arbeitsberichte, Neue Folge, 9*, Bonn.

Adobe Systems Inc. (2008): "Adobe – Audition 3."
URL `http://www.adobe.com/de/products/audition/`

Atal, B. S.; S. Hanauer (1971): "Speech analysis and synthesis by linear prediction of the speech wave." *J. Acoustic Society of America* 50, 637—655.

AT&T Intellectual Property (2007): "AT&T Labs Text-to-Speech: Frequently Asked Questions."
URL `http://www.research.att.com/~ttsweb/tts/faq.php#ProdNV`

Bachmann, A.; S. Breuer (2007): "Development of a BOSS unit selection module for tone languages." In: *SSW6-2007*, 166–171.

Bagshaw, P. (1998): "Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression." *Computer, Speech and Language* 12, 119–142.

Balestri, M.; A. Pacchiotti; S. Quazza; P. L. Salza; S. Sandri (1999): "Choose the best to modify the least: a new generation concatenative synthesis system." In: *Proceedings of the European Conference on Speech Communication and Technology (Budapest, Hungary)*, Bd. 5, 2291–2294.

Baumann, S.; J. Trouvain (2001): "On the Prosody of German Telephone Numbers." In: *Proceedings of Eurospeech*, 557–560.

Belhoula, K. (1996): *Ein regelbasiertes Verfahren zur maschinellen Graphem-nach-Phonem-Umsetzung von Eigennamen in der Sprachsynthese*. Düsseldorf: VDI Verlag.

Bennett, C. L.; A. Black (2003): "Using Acoustic Models to Choose Pronunciation Variations for Synthetic Voices." In: *Proceedings of Eurospeech*, Geneva, Switzerland, 2937–2940.
URL `citeseer.ist.psu.edu/684294.html`

Beutnagel, M.; A. Conkie; J. Schroeter; Y. Stylianou; A. Syrdal (1999): "ATT Next-Gen TTS System."
URL `citeseer.ist.psu.edu/beutnagel99nextgen.html`

Birkholz, P.; I. Steiner; S. Breuer (2007): "Control concepts for articulatory speech synthesis." In: *SSW6-2007*, 5–10.

Black, A.; K. Lenzo (2001): "Optimal Data Selection for Unit Selection Synthesis." In: *Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Pitlochry, Scotland, 63–67.

Black, A.; P. Taylor (1994): "CHATR: a generic speech synthesis system." In: *Proceedings of COLING-94*, Kyoto, Japan, Bd. II, 983–986.

Black, A. W.; N. Campbell (1995): "Optimising selection of units from speech databases for concatenative synthesis." In: *Proc. Eurospeech '95*, Madrid, Spain, 581–584.
URL `citeseer.ist.psu.edu/black95optimising.html`

Black, A. W.; K. A. Lenzo (2008): "Overview of Speech Synthesis."
URL `http://www.festvox.org/bsv/c59.html#AEN61`

Black, A. W.; P. Taylor (1997a): "Automatically Clustering Similar Units for Unit Selection in Speech Synthesis." In: *Proc. Eurospeech '97*, Rhodes, Greece, 601–604.
URL `citeseer.ist.psu.edu/black97automatically.html`

Black, A. W.; P. Taylor; R. Caley (1999): *The Festival Speech Synthesis System: System documentation*. CSTR, Edinburgh, edition 1.4 for festival version 1.4.0 Aufl.
URL `http://www.cstr.ed.ac.uk/projects/festival/manual/`

Black, A. W.; P. A. Taylor (1997b): *The Festival Speech Synthesis System: System Documentation. Techn. Ber. HCRC/TR-83*, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

Boersma, P.; D. Weenink (2001): "PRAAT, a system for doing phonetics by computer." *Glot International* 5(9/10), 341–345.

Bogert, B. P.; M. J. R. Healy; J. W. Tukey (1963): "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum, and Saphe Cracking." In: Rosenblatt, M., Hg., *Proceedings of the Symposium on Time Series Analysis*, New York: Wiley, 209–243.

BOSS (2005): "Sourceforge.net: Bonn Open Synthesis System (BOSS)."
URL `http://sourceforge.net/projects/boss-synth`

Bozkurt, B.; T. Dutoit; O. Ozturk (2003): "Text Design For TTS Speech Corpus Building Using A Modified Greedy Selection." In: *Proceedings of Eurospeech*, Geneva, Switzerland, 277–280.

Breen, A.; P. Jackson (1998): "Non-uniform unit selection and the similarity metric within BT's Laureate TTS system." In: *Proceedings of the Third ESCA/COCOSDA Workshop on Speech Synthesis*, Jenolan Caves, NSW, Australia, 201 – 206.

Breiman, L.; J. H. Friedman; R. A. Olshen; C. J. Stone (1984): *Classification and Regression Trees*. Belmont, California: Wadsworth International Group.

Breuer, S. (2000): "Reduktionsanalyse mit CART." In: Fellbaum, K., Hg., *Tagungsband Elektronische Sprachsignalverarbeitung (ESSV)*, Studientexte zur Sprachkommunikation, Cottbus.

Breuer, S.; J. Abresch (2003): "Unit Selection Speech Synthesis for a Directory Enquiries Service." In: *Proceedings of the ICPhS*, Barcelona.

Breuer, S.; J. Abresch (2004): "Phoxsy: Multi-phone Segments for Unit Selection Speech Synthesis." In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Jeju.

Breuer, S.; J. Abresch; P. Wagner; K. Stöber (2001): "BLF - ein Labelformat für die maschinelle Sprachsynthese mit BOSS II." In: Hess, W.; K. Stöber, Hg., *Tagungsband Elektronische Sprachsignalverarbeitung (ESSV)*, Studientexte zur Sprachkommunikation, Bonn.
URL `http://www.ikp.uni-bonn.de/publications/sbrjab_essv01.pdf`

Breuer, S.; S. Bergmann; R. Dragon; S. Möller (2006): "Set-up of a Unit-Selection Synthesis with a Prominent Voice." In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, Genova.

Breuer, S.; M. Holčík (2005): "boss_synthesis.cpp."
URL `boss/synthesis/boss_server/boss_synthesis.cpp`

Burnett, D. C.; M. R. Walker; A. Hunt (2004): "Speech Synthesis Markup Language (SSML) Version 1.0."
URL `http://www.w3.org/TR/speech-synthesis/`

Busser, B. (1998): "TreeTalk-D: A Machine Learning Approach to Dutch Word Pronunciation." In: *Text, Speech, Dialog*, 3–8.
URL `citeseer.ist.psu.edu/128132.html`

Busser, B.; W. Daelemans; A. van den Bosch (1999): "Machine learning of word pronunciation: the case against abstraction." In: *Proceedings of Eurospeech*, Budapest, 2123–2126.

Campbell, N. (1996): "CHATR: A High-Definition Speech Re-Sequencing System." *Acoustical Society of America and Acoustical Society of Japan, Third Joint Meeting* .

Campbell, N. (2001): "Building a corpus of natural speech - and tools for the processing of expressive speech." In: *Proceedings of Eurospeech*, 1525–1528.

Campbell, N. (2003): "Databases of Expressive Speech." In: *COCOSDA*.

Campbell, N.; A. Black (1996): "Prosody and the selection of units for concatenative synthesis." In: van Santen et al. (1996).

Campbell, W. N. (1992): "Syllable-based segmental duration." In: Bailly, G.; C. Benoît; T. R. Sawallis, Hg., *Talking machines: Theories, models, and designs*, Amsterdam: Elsevier Science, 211–224.

Cardelli, L.; P. Wegner (1985): "On Understanding Types, Data Abstraction, and Polymorphism." *ACM Computing Surveys* 17(4), 471–522.
URL `citeseer.ist.psu.edu/cardelli85understanding.html`

Carvalho, P.; I. Trancoso; L. Oliveira (2003): "WFST based Unit Selection for Concatenative Speech Synthesis in European Portuguese." In: *Proceedings of the 15th ICPhS*, Barcelona, Spain, 2333–2336.

Clark, J. (1999): "XSL Transformations (XSLT) Version 1.0."
URL `http://www.w3.org/TR/1999/REC-xslt-19991116`

Clark, R. A. J.; K. Richmond; S. King (2007): "Multisyn: Open-domain unit selection for the Festival speech synthesis system." *Speech Communication* 49(4), 317–330.

Code Sector Inc. (2008): "Audio Sliders – Windows Volume Control Replacement." URL `http://www.codesector.com/audiosliders.php`

Conkie, A. (1999): "Robust unit selection system for speech synthesis." URL `citeseer.ist.psu.edu/conkie99robust.html`

Conkie, A.; S. Isard (1996): "Optimal coupling of diphones." In: van Santen et al. (1996), 293—305.

Crowe, A.; M. Jack (1987): "Globally optimising formant tracker using generalised centroids." *Electronic Letters* 23(19), 1019–1020.

Daelemans, W. (1999): "Memory-Based Language Processing. Introduction to the Special Issue." *Journal of Experimental and Theoretical AI (JETAI)* 11, 3.

Daelemans, W.; A. van den Bosch (1996a): "Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion." In: Van Santen, J.; R. Sproat; J. Olive; J. Hirschberg, Hg., *Progress in Speech Synthesis*, New York: Springer Verlag, 77–79.

Daelemans, W.; A. van den Bosch (2001): "TreeTalk: Memory-Based Word Phonemisation." In: Damper, R. I., Hg., *Data-Driven Techniques in Speech Synthesis*, Kluwer Academic Publishers, 149–172.

Daelemans, W.; J. Zavrel; K. van der Sloot (2001): *TiMBL: Tilburg Memory Based Learner, version 4.0, Reference Guide. Technical Report 01-04*, ILK.

Daelemans, W.; J. Zavrel; K. van der Sloot; A. van den Bosch (2000): *TiMBL: Tilburg Memory Based Learner, version 3.0, Reference Guide. Techn. Ber.*, ILK Technical Report 00-01.
URL `http://ilk.kub.nl`

Daelemans, W. M.; A. P. J. van den Bosch (1996b): "Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion." In: van Santen et al. (1996), 77–89.

Damper, R. I.; Y. Marchand; M. J. Adamson; K. Gustafson (1999): "Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches." *Computer Speech and Language* 13, 155–176.

Dau, T.; D. Püschel; A. Kohlrausch (1996): "A quantitative model of the "effective" signal processing in the auditory system. II. Simulations and measurements." *Journal of the Acoustical Society of America* 99(6), 3623–3631.

Dettweiler, H.; W. J. Hess (1985): "Concatenation rules for demisyllable speech synthesis." *Acustica* 57, 268–283.

DFKI (2008): "The MARY Text-to-Speech System."
URL `http://mary.dfki.de/`

Donovan, R. (2000): "Segment pre-selection in decision-tree based speech synthesis systems." In: *Proceedings of ICASSP*, Istanbul, Turkey, Bd. 2, 937 – 940.

Donovan, R. (2001): "A New Distance Measure for Costing Spectral Discontinuities in Concatenative Speech Synthesisers." In: *Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Pitlochry, Scotland.
URL `citeseer.ist.psu.edu/donovan01new.html`

Donovan, R. E.; E. M. Eide (1998): "The IBM trainable speech synthesis system." In: *Proceedings of ICSLP*, Sydney, Australia.

Donovan, R. E.; P. C. Woodland (1999): "A hidden Markov-model-based trainable speech synthesizer." *Computer Speech and Language* 13, 223–241.

Dutoit, T. (1997): *An Introduction to Text-to-Speech Synthesis*, Bd. 3 von *Text, Speech and Language Technology*. Dordrecht: Springer.

ECESS (2008): "European Center of Excellence on Speech Synthesis."
URL `http://www.ecess.eu/`

Ellbogen, T.; F. Schiel; A. Steffen (2004): "The BITS Speech Synthesis Corpus for German." In: *Proceedings of LREC*, Lisbon, Portugal.

Elovitz, H. S.; R. Johnson; A. McHugh; J. E. Shore (1976): "Letter-to-sound rules for automatic translation of English text to phonetics." *IEEE Transactions on Acoustics, Speech and Signal Processing ASSP* 24, 446459.

Francois, H.; O. Boeffard (2001): "Design of an optimal continuous speech database for text-to-speech synthesis considered as a set covering problem." In: *Proceedings of Eurospeech*, 829–832.

Fraser, M.; S. King (2007): "The Blizzard Challenge 2007." In: *Proceeding of SSW6*, Bonn, Germany.

Fujisaki, H. (1988): "A note on the physiological and physical basis for the phrase and accent components in the voice fundamental frequency contour." In: Fujimura, O., Hg., *Vocal physiology: Voice production, mechanisms and functions*, New York: Raven, 347–355.

Gibbon, D. (1995): "SAMPA-D-VMlex. Dokumentation V1.0."
URL `http://coral.lili.uni-bielefeld.de/Documents/sampa-d-vmlex.html`

Goldsmith, J. (1999): "Dealing with Prosody in a Text to Speech system." *International Journal of Speech Technology* 3(1), 51–63.

Grice, M.; S. Baumann; R. Benzmüller (2005): "German Intonation in Autosegmental-Metrical Phonology." In: Jun, S.-A., Hg., *Prosodic Typology: The Phonology of Intonation and Phrasing*, Oxford University Press.

Grůber, M.; D. Tihelka; J. Matoušek (2007): "Evaluation of various unit types in the unit selection approach for the Czech language using the Festival system." In: *Proceedings of SSW6*, Bonn, Germany, 276–281.

Gustafson, J. (1996): *A Swedish Name Pronunciation System.* Licentiate thesis, TMH, KTH Stockholm.

Haapalainen, M.; A. Majorin (1995): "GERTWOL und Morphologische Disambiguierung für das Deutsche." In: *NODALIDA*.

Hammerstingl, R. (2003): *Evaluation eines Sprachsynthesesystems nach dem Prinzip der Nonuniform Unit Selection: Zur segmentalen Verständlichkeit von Namen.* Magisterarbeit, Universität Bonn.

Hammerstingl, R.; S. Breuer (2003): "Evaluation eines Sprachsynthesesystems nach dem Prinzip der Nonuniform Unit Selection." In: *Tagungsband Elektronische Sprachsignalverarbeitung (ESSV)*, Karlsruhe.

Hammerstingl, R.; S. Breuer (2004): "Evaluation eines Sprachsynthesesystems nach dem Prinzip der Non-Uniform Unit Selection." In: *IKP-Arbeitsberichte, Neue Folge, 10*, Bonn.

Hao, J.; L. Yi; J. Li; X. Lou (2006): "A Unified Totally-Data-Driven Prediction of Duration and Pause in TTS." In: *The 8th International Conference on Signal Processing.*

Hégaret, P. L.; R. Whitmer; L. Wood (2005): "Document Object Model (DOM)."
URL `http://www.w3.org/DOM/`

Hess, W. (2002): "Systeme der akustischen Mensch-Maschine-Kommunikation. Kapitel 3: Sprachsynthese. Vorlesungsskript."
URL `http://www.ifk.uni-bonn.de/lehre/magister/informationen-und-materialien-kopho/materialien-1/hess/mensch-maschine-kommunikation/sam_3.pdf`

Heuft, B.; T. Portele; F. Höfer; H. Meyer; M. Rauth (1995): "Betonungsstufen von Silben und ihre Beziehung zum Sprachsignal." In: *Fortschritte der Akustik - DAGA*, Saarbrücken, 999–1002.

Hirokawa, T. (1989): "Speech synthesis using a waveform dictionary." In: *Proceedings of Eurospeech*, 1140–1143.

Hirokawa, T.; K. Hakoda (1990): "Segment selection and pitch modification for high quality speech synthesis using waveform segments." In: *Proceedings of ICSLP*, 337–340.

Holčík, M. (2005): "boss_module.h."
URL `boss/libraries/boss_utility/boss_module.h`

Holčík, M.; S. Breuer (2005): "BOSS API Documentation."
URL `http://boss-synth.sourceforge.net/api-docs/`

Huang, X.; A. Acero; J. Adcock; H. Hon; J. Goldsmith; J. Liu; M. Plumpe (1996): "Whistler: A Trainable Text-to-Speech System." In: *Proc. ICSLP '96*, Philadelphia, PA, Bd. 4, 2387–2390.

Hunt, A. J.; A. W. Black (1996): "Unit selection in a concatenative speech synthesis system using a large speech database." In: *Proceedings of ICASSP*, Atlanta, Georgia, USA, Bd. 1, 373 – 376.

IfK (2008a): "BOMP."
URL `http://www.ifk.uni-bonn.de/forschung/abteilung-sprache-und-kommunikation/phonetik/sprachsynthese/bomp`

IfK (2008b): "Bonner Prosodische Datenbank."
URL `http://www.ifk.uni-bonn.de/forschung/abteilung-sprache-und-kommunikation/phonetik/prosodiegenerierung-fur-die-sprachsynthese/bonner-prosodische-datenbank`

ISO (2002): "ISO 639-1:2002 Codes for the representation of names of languages – Part 1: Alpha-2 code."

URL `http://www.iso.org/iso/en/prods-services/popstds/languagecodes.html`

Itakura, F.; S. Saito (1968): "An analysis-synthesis telephony based on maximum likelihood method." In: *Proc. Int. Cong. Acoust.*, Bd. C-5-5, C17–20.

Iwahashi, N.; N. Kaiki; Y. Sagisaka (1992): "Concatenative speech synthesis by minimum distortion criteria." In: *Proc. ICASSP*, San Francisco, Bd. 2, 65–68.

Iwahashi, N.; Y. Sagisaka (1992): "Speech segment network approach for an optimal synthesis unit set." In: *ICSLP*, 479–482.

Jannedy, S.; B. Möbius (1997): "Name pronunciation in German text-to-speech synthesis." In: *Proceedings of ANLP*, 49–56.

Karger, R.; W. Wahlster (2000): "Facts and Figures about the Verbmobil Project." In: Wahlster (2000), 22–30.

Karlsson, F. (1990): *SWETWOL: A Comprehensive Morphological Analyser For SwedishDepartment of General Linguistics. Manuscript*, Helsinki.

Kindermann, J. (2005): "PiAvida."
URL `http://www.ais.fraunhofer.de/tp/piavida.html`

Kirstein, M.; D. Stock (1976): *Amplituden- und intervallstatistische Messungen an Sprachsignalen*, Bd. 54 von *Forschungsberichte des Instituts für Kommunikationsforschung und Phonetik der Universität Bonn*. Buske.

Klabbers, E.; K. Stöber; R. Veldhuis; P. Wagner; S. Breuer (2001): "Speech synthesis development made easy: The Bonn Open Synthesis System." In: *Proceedings of Eurospeech*, Aalborg.

Klabbers, E.; R. Veldhuis (1998): "On the Reduction of Concatenation Artefacts in Diphone Synthesis." In: *Proceedings of ICSLP*, Sydney, Australia.
URL `citeseer.ist.psu.edu/klabbers98reduction.html`

Klabbers, E. A. M.; K. Stöber (2001): "Creation of speech corpora for the multilingual Bonn Open Synthesis System." In: *Proc. of 4th ESCA Workshop on Speech Synthesis*, Pitlochry, Scotland, 23–28.

Klatt, D. (1987): "Review of text-to-speech conversion for English." *J. Acous. Soc. Amer.* 82, 737–793.
URL `http://americanhistory.si.edu/archives/speechsynthesis/dk_737a.htm`

Koskenniemi, K. (1983): "Two-level Model for Morphological Analysis." In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe.

Kraft, V. (1997): *Verkettung natürlichsprachlicher Bausteine zur Sprachsynthese: Anforderungen, Techniken und Evaluierung.* Dissertation, Institut für Kommunikationsakustik, Ruhr-Universität Bochum., Düsseldorf.

KTH Stockholm (2002): "ESPS."
URL `http://www.speech.kth.se/speech/esps/esps.zip`

Kullback, S.; R. A. Leibler (1951): "On information and sufficiency." *Annals of Mathematical Statistics* 22(1), 79–86.

Küpfmüller, K.; O. Warns (1956): "Sprachsynthese aus Lauten." *Nachrichtentechnische Fachberichte* 3, 28–31.

Lafferty, J.; A. McCallum; F. Pereira (2001): "Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data." In: *Proceedings of ICML*.

LaTeX (2008): "LaTeX project: LaTeX – A document preparation system."
URL `http://www.latex-project.org/`

Macchi, M.; M. Spiegel (1990): "Using a demisyllable inventory to synthesize names." *Speech Technology* , 208–212.

Mahalanobis, P. (1936): "On the generalized distance in statistics." *Proceedings of the National Institute of Science of India* 12, 49–55.

Mangold, M., Hg. (1990): *Duden Aussprachewörterbuch*, Bd. 6 von *Der Duden: in 10 Bänden; das Standardwerk zur deutschen Sprache.* Mannheim, Wien, Zürich: Dudenverlag, 3. Aufl.

Marchand, Y.; R. I. Damper (2000): "A multi-strategy approach to improving pronunciation by analogy." *Computational Linguistics* 26, ing pronunciation by analogy. Computational Linguistics, 26:1.

Mengel, A. (1994): "Four approaches to automatic transcription of German names. A comparison.", vortrag auf dem Onomastica Research Colloquium in London.

Menzerath, P.; A. de Lacerda (1933): *Koartikulation, Steuerung und Lautabgrenzung.* Berlin, Bonn.

Mermelstein, P. (1976): "Distance measures for speech recognition, psychological and instrumental." In: Chen, C. H., Hg., *Pattern Recognition and Artificial Intelligence*, New York: Academic, 374–388.

Mixdorff, H. (1998): *Intonation Patterns of German - Model-based Quantitative Analysis and Synthesis of F0-Contours*. Dissertation, TU Dresden.
URL `http://www.tfh-berlin.de/~mixdorff/thesis/index.html`

Möbius, B. (1993): *Ein quantitatives Modell der deutschen Intonation—Analyse und Synthese von Grundfrequenzverläufen*. Nr. 305 in Linguistische Arbeiten, Tübingen: Max Niemeyer Verlag.

Möbius, B. (1998): "Word and syllable models for German text-to-speech synthesis." In: *Proceedings of the Third International Workshop on Speech Synthesis*, enolan Caves, Australia, 59–64.

Möbius, B. (2000): "Corpus-based speech synthesis: methods and challenges." *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (Univ. Stuttgart), AIMS* 6(4), 87–116.

Möbius, B. (2003): "Rare events and closed domains: Two delicate concepts in speech synthesis." *International Journal of Speech Technology* 6(1), 57–71.

Moers, D.; P. Wagner; S. Breuer (2007): "Assessing the adequate treatment of fast speech in unit selection speech synthesis systems for the visually impaired." In: *SSW6-2007*, 282–287.

Möhler, G.; A. Schweitzer; M. Breitenbücher (2007): "Experimental Phonetics - Synthesis at IMS."
URL `http://www.ims.uni-stuttgart.de/phonetik/synthesis/festival_opensource.html`

Moulines, E.; F. Charpentier (1990): "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones." *Speech Commun.* 9(5-6), 453–467.

MySQL AB (2008): "MySQL AB :: MySQL Documentation."
URL `http://dev.mysql.com/doc/`

Object Modeling Group (2007): "UML 2.1.2."
URL `http://www.omg.org/spec/UML/2.1.2/`

Onomastica, F. R. (1995): *ONOMASTICA Multi-Language Pronunciation Dictionary of Proper names and Place Names*. Project No. LRE-61004.

Ortmann, W. D. (1993): *Kernmorpheme im Deutschen. 4561 allgemeinsprachliche deutsche Morphemgestalten im Vergleich zwischen zehn Quellen*. Programmierung und EDV-Arbeiten, Goethe-Institut München.

O'Shaughnessy, D. (1987): *Speech Communication. Human and machine*. New York: IEEE Press.

Pérez, J.; A. Bonafonte; H.-U. Hain; E. Keller; S. Breuer; J. Tian (2006): "ECESS Inter-Module Interface Specification for Speech Synthesis." In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, Genova.

Peterson, G. E.; W. Wang; E. Sievertsen (1958): "Segmentation techniques in speech synthesis." *Journal of the Acoustical Society of America* 30, 739–742.

Portele, T. (1996): *Ein phonetisch-akustisch motiviertes Inventar zur Sprachsynthese deutscher Äußerungen*. Tübingen: Niemeyer.

Portele, T. (1998): "JUst CONcatenation - A Corpus-based Approach and its Limits." In: *Proceedings of SSW3*, 153–158.

Portele, T.; J. Krämer; D. Stock (1995): "Symbolverarbeitung im Sprachsynthesesystem Hadifix." In: *Proc. 6. Konferenz Elektronische Sprachsignalverarbeitung*, Wolfenbüttel, 97–104.

Prince, A.; P. Smolensky (2004): *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishers.
URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/1405119330`

Prudon, R. (2002): *Synthèse de la parole multilocuteur par sélection d'unités acoustiques*. Dissertation, thèse de l'Université Paris XI, Orsay.

Rabiner, L. R. (1989): "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* 77(2), 257–286.

Riley, M. (1991): "A statistical model for generating pronunciation networks." In: *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*.

Rohde, H.; S. Breuer (2005): "An HNM-Synthesizer for BOSS." In: *Proceedings of the 16th conference on Electronic Speech Signal Processing (ESSP).*, Prague.

Sagisaka, Y. (1988): "Speech synthesis by rule using an optimal selection of non-uniform synthesis units." In: *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, New York: IEEE, Bd. 1, 679 – 682.

Sagisaka, Y. (1989): "On the unit set design for speech synthesis by. rule using nonuniform units." *ASA Fall Meeting FF24, JASA* 1(86 Suppll S79), S79.

Sagisaka, Y.; N. Kaiki; N. Iwahashi; K. Mimura (1992): "ATR ν-talk speech synthesis system." In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 483–486.

Salomaa, A.; M. Soittola (1978): *Automata-Theoretic Aspects of Formal Power Series*. New York: Springer.

van Santen, J. (1997): "Combinatorial issues in text-to-speech synthesis." In: *Proceedings of Eurospeech*, Rhodes, Greece.

van Santen, J.; A. L. Buchsbaum (1997): "Methods for optimal text selection." In: *Proceedings of Eurospeech*, Rhodes, Greece.

van Santen, J.; R. Sproat; J. Olive; J. Hirschberg, Hg. (1996): *Progress in Speech Synthesis*. New York: Springer.

Schmidt, M.; S. Fitt; C. Scott; M. Jack (1993): "Phonetic transcription standards for European names (ONOMASTICA)." In: *Proceedings: Eurospeech*, Bd. 1, 279–82.

Schröder, M.; S. Breuer (2004): "XML Representation Languages as a Way of Interconnecting TTS Modules." In: *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Jeju.

Schröder, M.; J. Trouvain (2003): "The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching." *International Journal of Speech Technology* 6, 365–377.

Schröder, T. (2004): *Instrumentelle Qualitätsmaße in der Sprachsynthese: Amplituden- und intervallstatistische Untersuchungen*. Diplomarbeit, Universität Bonn.

Schweitzer, A.; N. Braunschweiler; T. Klankert; B. Möbius; B. Säuberlich (2003): "Restricted Unlimited Domain Synthesis." In: *Proceedings of Eurospeech*, Geneva, Switzerland, 1321–1324.

Schwill, D. (2001): *Untersuchung prosodischer Parameter bei Phrasenfinalität*. Magisterarbeit, IKP, Universität Bonn.

Sejnowski, T. J.; C. R. Rosenberg (1987): "Parallel networks that learn to pronounce English text." *Complex Systems* 1, 145168.

Sjölander, K.; J. Beskow (2000): "WaveSurfer - an open source speech tool." In: *Proc of ICSLP*, Beijing, Bd. 4, 464–467.

Sproat, R., Hg. (1998): *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Dordrecht: Kluwer Academic.

Sproat, R.; A. Hunt; M. Ostendorf; P. Taylor; A. Black; K. Lenzo; M. Edgington (1998): "SABLE: A STANDARD FOR TTS MARKUP."
URL `http://www.bell-labs.com/project/tts/sabpap/sabpap.html`

Stöber, K. (2002): *Bestimmung und Auswahl von Zeitbereichseinheiten für die konkatenative Sprachsynthese*. Frankfurt/M.: P. Lang.

Stöber, K.; J. Bröggelwirth (2000): "boss_morphemes_de.cpp."
URL `boss/libraries/boss_transcription/boss_morphemes_de.cpp`

Stöber, K.; T. Portele; P. Wagner; W. Hess (1999): "Synthesis by Word Concatenation." In: *Proceedings of Eurospeech*, Budapest, Bd. 2, 619–622.

Stöber, K.; P. Wagner; J. Helbig; S. Köster; D. Stall; M. Thomae; J. Blauert; W. Hess; R. Hoffmann; H. Mangold (2000): *Speech Synthesis Using Multilevel Selection and Concatenation of Units from Large Speech Corpora*. In: Wahlster (2000).

Stöber, K.; P. Wagner; E. Klabbers; W. Hess (2001): "Definition of a Training Set for Unit Selection-Bases Speech Synthesis." In: *Proceedings of 4th ISCA Workshop on Speech Synthesis*, Pitlochry, Scotland.

Stock, D. (1992): "P-TRA - eine Programmiersprache zur phonetischen Transkription." In: Hess, W.; W. Sendlmeier, Hg., *Beiträge zur angewandten und experimentellen Phonetik. Beiheft 72 der Zeitschrift für Dialektologie und Linguistik*, Stuttgart: Steiner, 222–231.

Stylianou, Y. (2000): "On the Implementation of the Harmonic Plus Noise Model for Concatenative Speech Synthesis." In: *Proceedings of ICASSP*, Istanbul, Turkey.

Syrdal, A. K.; A. D. Conkie (2005): "Perceptually-based Data-driven Join Costs: Comparing Join Types." In: *Proceedings of Interspeech*, Lisbon, Portugal.

Takeda, K.; K. Abe; Y. Sagisaka (1990): "On the Unit Search Criteria and Algorithms for Speech Synthesis Using Non-Uniform Units." In: *Proceedings of the ICSLP*, 341–344.

Tamburini, F.; P. Wagner (2007): "On Automatic Prominence Detection for German." In: *Proc. InterSpeech 2007*, Antwerp, 1809–1812.

Taylor, P. (2007): *Text-to-Speech Synthesis. Draft of 7th February 2007.*

Taylor, P.; A. W. Black (1999): "Speech synthesis by phonological structure matching." In: *Proceedings of the European Conference on Speech Communication and Technology (Budapest, Hungary)*, Bd. 2, 623–626.

Taylor, P.; A. Isard (1997): "SSML: A speech synthesis markup language." *Speech Communication* (21), 123–133.
URL `citeseer.ist.psu.edu/taylor97ssml.html`

Taylor, P. A. (2000): "Concept-to-speech synthesis by phonological structure matching." *Philosophical Transactions of The Royal Society, series A* 358(1769), 1403–1417.

Taylor, P. A.; A. Black; R. Caley (1998): "The architecture of the Festival speech synthesis system." In: *Proceedings of the Third ESCA Workshop in Speech Synthesis*, Jenolan Caves, Australia, 147–151.

Tcl/TK (2008): "Tcl Developer Site."
URL `http://www.tcl.tk/`

The Apache Software Foundation (2007): "Xerces C++ Parser."
URL `http://xerces.apache.org/index.html`

The Free Software Foundation (2008): "GNU Libtool."
URL `http://www.gnu.org/software/libtool/`

The Perl Foundation (2008): "The Perl Directory - perl.org."
URL `http://www.perl.org/`

Tokuda, K.; H. Zen; A. Black (2004): "An HMM-based approach to multilingual speech synthesis." In: Narayanan, S.; A. Alwan, Hg., *Text to Speech Synthesis: New Paradigms and Advances*, New Jersey: Prentice Hall, 135–153.

Trolltech (2008): "Qt — Trolltech."
URL `http://trolltech.com/products/qt/homepage`

Vaughan, G. V.; B. Elliston; T. Tromey; I. L. Taylor (2000): "GNU Autoconf, Automake and Libtool."
URL `http://sources.redhat.com/autobook/`

Vepa, J.; S. King; P. Taylor (2002): "New Objective Distance Measures for Spectral Discontinuities in Concatenative Speech Synthesis." In: *Proc. IEEE 2002 workshop on speech synthesis*, Santa Monica, USA.

Vitale, A. (1991): "An algorithm for high-accuracy name pronunciation by parametric speech synthesizer." *Computational Linguistics* 17, 257–276.

Viterbi, A. (1967): "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Transactions on Information Theory* 13(2), 260–269.

Wagner, P.; S. Breuer; K. Stöber (2000): "Automatische Prominenzetikettierung einer Datenbank für die korpusbasierte Sprachsynthese." In: *DAGA - Fortschritte der Akustik*, Oldenburg.
URL `http://www.ikp.uni-bonn.de/publications/pwa_daga00.ps.gz`

Wagner, P.; F. Haas; K. Stöber; J. Helbig (1999): "Multilinguale korpusbasierte Sprachsynthese auf der Basis domänenspezifischen Ausgangsmaterials." In: Mehnert, D., Hg., *Tagungsband Elektronische Sprachsignalverarbeitung ESSV'1999*, Görlitz, Bd. 16 von *Studientexte zur Sprachkommunikation*, 152–159.

Wahlster, W., Hg. (2000): *Verbmobil: Foundations of Speech-to-Speech Translation*. Berlin: Springer.

Weiss, C. (2007): *Adaptive audio-visuelle Synthese: Automatische Trainingsverfahren für Unit-Selection-basierte audio-visuelle Sprachsynthese*. Dissertation, Universität Bonn.
URL `http://hss.ulb.uni-bonn.de/diss_online/phil_fak/2007/weiss_christian`

Weiss, C.; M. Da Silva; K. Tokuda; W. Hess (2005): "Low Resource HMM-based Speech Synthesis applied to German." In: Hoffmann, R., Hg., *Proceedings 16th Conference ,,Elektronische Sprachsignalverarbeitung" (ESSP 2005)*, Prague.

Weiss, C.; W. Hess (2006): "Conditional Random Fields for Hierarchical Segment Selection in Text-to-Speech Synthesis." In: *Proc. Interspeech 2006*, Pittsburgh, PA, USA.

Wells, J. (1997): "SAMPA computer readable phonetic alphabet." In: Gibbon, D.; R. Moore; R. Winski, Hg., *Handbook of Standards and Resources for Spoken Language Systems*, Berlin and New York: Mouton de Gruyter, Kap. Part IV, section B.

Wells, J. C. (2005): "SAMPA computer readable phonetic alphabet." URL `http://www.phon.ucl.ac.uk/home/sampa/index.html`

Zen, H.; T. Nose; J. Yamagishi; S. Sako; T. Masuko; A. Black; K. Tokuda (2007): "The HMM-based speech synthesis system version 2.0." In: *Proc. of ISCA SSW6*, Bonn, Germany.

Anhang A. BOSS-SAMPA/phoxsy

A.1. Zusammensetzung der Einheiten für das Deutsche

In diesem Abschnitt werden die Basiseinheiten der Phoxsy-Multiphone (Kapitel 3) näher beschrieben. Eine vollständige Liste der tatsächlich verwendeten Einheiten findet sich in Tabelle A.2.1.

A.1.1. Vokale (Monophthonge)

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
iː	i:	i:	Biene
ɪ	I	I	Bitte
yː	y:	y:	Blüte
ʏ	Y	Y	Mütter
eː	e:	e:	Regen
ɛ	E	E	Menge
ɛː	E:	E:	Räte
ø	2:	2:	Röte
œ	9	9	köstlich
ə	@	@	Bitte
ɐ	6	6	bitter
a	a	a	Watte
aː	a:	a:	waten
ɔ	O	O	Wolle
oː	o:	o:	Wohl
ʊ	U	U	rund
uː	u:	u:	Buch

A.1.2. Nasalierte Vokale

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
ɛ̃	E~	E~1	Timbre
ɔ̃	O~	O~1	Bonbon
œ̃	9~	9~1	Parfum
ɑ̃	A~	a~	Restaurant

A.1.3. Diphthonge

Zusätzlich zu den üblicherweise als Diphthong beschriebenen Einheiten werden auch Vokale mit auf sie folgenden /r/-Allophonen der selben Silbe zu einer Einheit verschmolzen und als Diphthonge etikettiert.

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
aɪ	aI	aI	Seite
aʊ	aU	aU	Laut
ɔʏ	OY	OY	Feuer
iːɐ	i:6	i:6	vier
ɪɐ	I6	I6	Hirte
yːɐ	y:6	y:6	Tür
ʏɐ	Y6	Y6	Würde
eːɐ	e:6	e:6	Wer
ɛɐ	E6	E6	Härte
ɛːɐ	E:6	E:6	Bär (auch e:6)
øːɐ	2:6	2:6	Stör
œɐ	96	96	Hörner
aːɐ	a:6	a:6	Haar
aɐ	a6	a6	hart
uːɐ	u:6	u:6	Flur
ʊɐ	U6	U6	kurz
oːɐ	o:6	o:6	Rohr
ɔɐ	O6	O6	Morgen

A.1.4. Konsonanten im Anlaut

Die hier aufgeführten Konsonanten werden stets mit ihren unmittelbaren Folgevokalen zu einer phoxsy-Einheit verbunden.

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
j/ʝ/ç/i̯	j	j / j\[1] / C / <i	Jahr / Ach ja
l	l	l	Lauge / Wald
ʁ/ʀ/ʁ̞/ʁ̥(χ)	r	R / R\/[1] R_o[1] / R_o_O[1] (X)	Raute / Raute / Waren / treten
h/ɦ	h	h / h\[1]	Hand
ʊ/v	v	v / P[1]	wann
ɹ	R	r	Roadmap
w/o̯/ɥ	w	w/[1] / <u / <o	Web / Reservoir / Etui
ʔ	?	? (Q)	_Apfel

A.1.5. Approximant/Liquid + Schwa + Konsonant

Diese Einheiten decken Fälle ab, in denen Suffixe häufig zu silbischen Konsonanten reduziert werden, die aber ebenso als Folge von Vokal und Konsonant realisiert werden können. Weitere, nicht gelistete, Einheiten entstehen durch die Verbindung der Anlautkonsonanten mit diesen Einheiten. Zwar sind von diesen nur wenige von praktischer Bedeutung, dennoch wurden sicherheitshalber alle Kombinationen in das deutsche Inventar aufgenommen.

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
əl/l̩	@l	l=	Rubel
ən/n̩	@n	@n, n=	raten / raten
əm/m̩	@m	@m,m=	jedem/Raben

A.1.6. Nasale

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
m	m	m	Mund
n	n	n	Nadel
ŋ	N	N	Inge

A.1.7. Frikative

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
f	f	f	fahren
s	s	s	Hass
z	z	z	Hase
ʃ	S	S	waschen
ʒ	Z	Z	Garage
ç	C	C	Pfirsich
x/χ	x	x / X[1]	Woche / Wache

A.1.8. Plosive

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
p/pʰ	p	p / p_h[1]	spitz / Pause
b/b̥	b	b / b_0[1]	laben / Braten
t/tʰ	t	t	Station / Taufe
d/d̥	d	d	Adel / Drang
k/kʰ	k	k	Skat / Kammer
g/g̊	g	g	Agathe / Grat

A.1.9. Affrikaten

IPA	BOSS-SAMPA/ phoxsy (DE)	SAMPA-D	Beispiel
pf	`pf`	`pf /p_f`[1]	Pfand
ts	`ts`	`ts / t_s`[1]	Katze
ʧ	`tS`	`tS`	Matsch
dʒ	`dZ`	`dZ`	Dschungel

[1]Notation in X-SAMPA (`http://www.phon.ucl.ac.uk/home/sampa/x-sampa.htm`)

A.2. Inventardefinition und Kontextklassen

Im Folgenden werden die in der deutschen Inventardefinition enthaltenen BOSS-SAMPA/phoxsy-Einheiten (s. Kapitel 3) mit ihren dazugehörigen Kontextklassen (s. Abschnitt 3.2.3) dargestellt. Zunächst werden dazu die in der Definition der zweiten Kontextklassenstufe verwendeten Abkürzungen der Artikulationsorte erläutert.

BAC	back (Hinterzungenvokal)
FRO	front (Vorderzungenvokal)
CEN	central (Mittelzungenvokal)
BIL	bilabialer Konsonant
LAB	labiodentaler Konsonant
DEN	dentaler Konsonant
ALV	alveolarer Konsonant
RET	retroflexer Konsonant
PAL	palataler Konsonant
VEL	velarer Konsonant
UVU	uvularer Konsonant

Die in der untenstehenden Tabelle der Einheiten und Kontextklassen mit § präfigierten Einheiten stellen Buchstabier- bzw. Zahleneinheiten dar, die jeweils komplette Wortrealisierungen repräsentieren. $p ist das in BOSS verwendete Symbol für eine Sprechpause. Die Spaltenüberschriften orientieren sich an den Attributnamen in den XML- und SQL-Definitionen für BOSS (s. Anhang C und D). So steht z. B. TKey für die Transkription einer Halbphon-, Phon-, Wort- oder Silbeneinheit. Hier findet dieses Attribut nur zur Beschreibung der Einheitentranskriptionen Anwendung.

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
2:	2	2	FRO	FRO	9	9	9	FRO	FRO
9:	9	9	FRO	FRO	6	6	6	CEN	CEN
@	@	@	CEN	CEN	a	a	a	CEN	CEN
A	A	A	CEN	CEN	a:	a	a	CEN	CEN
e:	e	e	FRO	FRO	E	E	E	FRO	FRO
E	E	E	FRO	FRO	E:	E	E	FRO	FRO

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
I	I	I	FRO	FRO	i:	i	i	FRO	FRO
o:	o	o	BAC	BAC	O	O	O	BAC	BAC
O	O	O	BAC	BAC	O:	O	O	BAC	BAC
u:	u	u	BAC	BAC	U	U	U	BAC	BAC
Y	Y	Y	FRO	FRO	y:	y	y	FRO	FRO
aI	aI	a	FRO	CEN	aU	aU	a	BAC	CEN
OY	OY	O	FRO	BAC	eI	eI	E	FRO	FRO
oU	oU	oU	BAC	BAC	@n	n	@n	ALV	ALV
@l	l	@l	ALV	ALV	@m	m	@m	BIL	BIL
r@n	n	r	ALV	UVU	r@l	l	r	ALV	UVU
r@m	m	r	BIL	UVU	l@n	n	l	ALV	ALV
l@l	l	l	ALV	ALV	l@m	m	l	BIL	ALV
v@n	n	v	ALV	LAB	v@l	l	v	ALV	LAB
v@m	m	v	BIL	LAB	w@n	n	w	ALV	BIL
w@l	l	w	ALV	BIL	w@m	m	w	BIL	BIL
?@n	n	?@	VEL	CEN	?@l	l	?@	VEL	CEN
?@m	m	?@	BIL	CEN	b	b	b	BIL	BIL
d	d	d	ALV	ALV	g	g	g	VEL	VEL
k	k	k	VEL	VEL	p	p	p	BIL	BIL
t	t	t	ALV	ALV	C	C	C	PAL	PAL
f	f	f	LAB	LAB	S	S	S	ALV	ALV
s	s	s	ALV	ALV	T	T	T	DEN	DEN
D	D	D	DEN	DEN	v	v	v	LAB	LAB
x	x	x	VEL	VEL	z	z	z	DEN	DEN
Z	Z	Z	ALV	ALV	l	l	l	ALV	ALV
m	m	m	BIL	BIL	N	N	N	VEL	VEL
n	n	n	ALV	ALV	pf	f	p	LAB	BIL
ts	s	t	ALV	ALV	tS	S	t	ALV	ALV
dZ	Z	d	ALV	ALV	2:6	6	2	CEN	FRO

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
96	6	9	CEN	FRO	a6	6	a	CEN	CEN
a:6	6	a	CEN	CEN	e:6	6	e	CEN	FRO
E6	6	E	CEN	FRO	E:6	6	E	CEN	FRO
I6	6	I	CEN	FRO	i:6	6	i	CEN	FRO
o:6	6	o	CEN	BAC	O6	6	O	CEN	BAC
O:6	6	O	CEN	BAC	u:6	6	u	CEN	BAC
U6	6	U	CEN	BAC	Y6	6	Y	CEN	FRO
y:6	6	y	CEN	FRO	?2:	2	?2	FRO	FRO
?9	9	?9	FRO	FRO	?9:	9	?9	FRO	FRO
?6	6	?6	CEN	CEN	?@	@	?@	CEN	CEN
?a	a	?a	CEN	CEN	?A	A	?A	CEN	CEN
?a:	a	?a	CEN	CEN	?aI	aI	?a	FRO	CEN
?aU	aU	?a	BAC	CEN	?eI	eI	?E	FRO	FRO
?oU	oU	?o	BAC	BAC	?e:	e	?e	FRO	FRO
?E	E	?E	FRO	FRO	?E:	E	?E	FRO	FRO
?E	E	?E	FRO	FRO	?I	I	?I	FRO	FRO
?i:	i	?i	FRO	FRO	?o:	o	?o	BAC	BAC
?O	O	?O	BAC	BAC	?O	O	?O	BAC	BAC
?O:	O	?O	BAC	BAC	?OY	OY	?O	FRO	BAC
?u:	u	?u	FRO	BAC	?U	U	?U	FRO	BAC
?Y	Y	?Y	FRO	FRO	?y:	y	?y	FRO	FRO
?2:6	6	?2	CEN	FRO	?96	6	?9	CEN	FRO
?a6	6	?a	CEN	CEN	?a:6	6	?a	CEN	CEN
?e:6	6	?e	CEN	FRO	?E6	6	?E	CEN	FRO
?E:6	6	?E	CEN	FRO	?I6	6	?I	CEN	FRO
?i:6	6	?i	CEN	FRO	?o:6	6	?o	CEN	BAC
?O6	6	?O	CEN	BAC	?O:6	6	?O	CEN	BAC
?u:6	6	?u	CEN	BAC	?U6	6	?U	CEN	BAC
?Y6	6	?Y	CEN	FRO	?y:6	6	?y	CEN	FRO

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
j2:	2	j	FRO	PAL	j9	9	j	FRO	PAL
j9:	9	j	FRO	PAL	j6	6	j	CEN	PAL
j@	@	j	CEN	PAL	j@n	n	j	ALV	PAL
j@l	l	j	ALV	PAL	j@m	m	j	BIL	PAL
ja	a	j	CEN	PAL	jA	A	j	CEN	PAL
ja:	a	j	CEN	PAL	jaI	aI	j	FRO	PAL
jaU	aU	j	BAC	PAL	jeI	eI	j	FRO	PAL
joU	oU	j	BAC	PAL	je:	e	j	FRO	PAL
jE	E	j	FRO	PAL	jE	E	j	FRO	PAL
jE:	E	j	FRO	PAL	jI	I	j	FRO	PAL
ji:	i	j	FRO	PAL	jo:	o	j	BAC	PAL
jO	O	j	BAC	PAL	jO	O	j	BAC	PAL
jO:	O	j	BAC	PAL	jOY	OY	j	FRO	PAL
ju:	u	j	BAC	PAL	jU	U	j	BAC	PAL
jY	Y	j	FRO	PAL	jy:	y	j	FRO	PAL
j2:6	6	j	CEN	PAL	j96	6	j	CEN	PAL
ja6	6	j	CEN	PAL	ja:6	6	j	CEN	PAL
je:6	6	j	CEN	PAL	jE6	6	j	CEN	PAL
jE:6	6	j	CEN	PAL	jI6	6	j	CEN	PAL
ji:6	6	j	CEN	PAL	jo:6	6	j	CEN	PAL
jO6	6	j	CEN	PAL	jO:6	6	j	CEN	PAL
ju:6	6	j	CEN	PAL	jU6	6	j	CEN	PAL
jY6	6	j	CEN	PAL	jy:6	6	j	CEN	PAL
l2:	2	l	FRO	ALV	l9	9	l	FRO	ALV
l9:	9	l	FRO	ALV	l6	6	l	CEN	ALV
l@	@	l	CEN	ALV	la	a	l	CEN	ALV
lA	A	l	CEN	ALV	la:	a	l	CEN	ALV
laI	aI	l	FRO	ALV	laU	aU	l	BAC	ALV
leI	eI	l	FRO	ALV	loU	oU	l	BAC	ALV

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
le:	e	l	FRO	ALV	lE	E	l	FRO	ALV
lE	E	l	FRO	ALV	lE:	E	l	FRO	ALV
lI	I	l	FRO	ALV	li:	i	l	FRO	ALV
lo:	o	l	BAC	ALV	lO	O	l	BAC	ALV
lO	O	l	BAC	ALV	lO:	O	l	BAC	ALV
lOY	OY	l	FRO	ALV	lu:	u	l	BAC	ALV
lU	U	l	BAC	ALV	lY	Y	l	FRO	ALV
ly:	y	l	FRO	ALV	l2:6	6	l	CEN	ALV
l96	6	l	CEN	ALV	la6	6	l	CEN	ALV
la:6	6	l	CEN	ALV	le:6	6	l	CEN	ALV
lE6	6	l	CEN	ALV	lE:6	6	l	CEN	ALV
lI6	6	l	CEN	ALV	li:6	6	l	CEN	ALV
lo:6	6	l	CEN	ALV	lO6	6	l	CEN	ALV
lO:6	6	l	CEN	ALV	lu:6	6	l	CEN	ALV
lU6	6	l	CEN	ALV	lY6	6	l	CEN	ALV
ly:6	6	l	CEN	ALV	r2:	2	r	FRO	UVU
r9	9	r	FRO	UVU	r9:	9	r	FRO	UVU
r6	6	r	CEN	UVU	r@	@	r	CEN	UVU
ra	a	r	CEN	UVU	rA	A	r	CEN	UVU
ra:	a	r	CEN	UVU	raI	aI	r	FRO	UVU
raU	aU	r	BAC	UVU	reI	eI	r	FRO	UVU
roU	ou	r	BAC	UVU	re:	e	r	FRO	UVU
rE	E	r	FRO	UVU	rE	E	r	FRO	UVU
rE:	E	r	FRO	UVU	rI	I	r	FRO	UVU
ri:	i	r	FRO	UVU	ro:	o	r	BAC	UVU
rO	O	r	BAC	UVU	rO	O	r	BAC	UVU
rO:	O	r	BAC	UVU	rOY	OY	r	FRO	UVU
ru:	u	r	BAC	UVU	rU	U	r	BAC	UVU
rY	Y	r	FRO	UVU	ry:	y	r	FRO	UVU

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
r2:6	6	r	CEN	UVU	r96	6	r	CEN	UVU
ra6	6	r	CEN	UVU	ra:6	6	r	CEN	UVU
re:6	6	r	CEN	UVU	rE6	6	r	CEN	UVU
rE:6	6	r	CEN	UVU	rI6	6	r	CEN	UVU
ri:6	6	r	CEN	UVU	ro:6	6	r	CEN	UVU
rO6	6	r	CEN	UVU	rO:6	6	r	CEN	UVU
ru:6	6	r	CEN	UVU	rU6	6	r	CEN	UVU
rY6	6	r	CEN	UVU	ry:6	6	r	CEN	UVU
R9	9	R	FRO	RET	R9:	9	R	FRO	RET
R6	6	R	CEN	RET	R@	@	R	CEN	RET
Ra	a	R	CEN	RET	Ra:	a	R	CEN	RET
RE	E	R	FRO	RET	RE:	E	R	FRO	RET
RI	I	R	FRO	RET	Ri:	i	R	FRO	RET
RO	O	R	BAC	RET	RO:	O	R	BAC	RET
Ru:	u	R	BAC	RET	RU	U	R	BAC	RET
RaI	aI	R	FRO	RET	RaU	aU	R	BAC	RET
ROY	OY	R	FRO	RET	ReI	eI	R	FRO	RET
RoU	oU	R	BAC	RET	R@n	n	R	ALV	RET
R@l	l	R	ALV	RET	R@m	m	R	BIL	RET
R96	6	R	CEN	RET	RE6	6	R	CEN	RET
RE:6	6	R	CEN	RET	RI6	6	R	CEN	RET
RO6	6	R	CEN	RET	RO:6	6	R	CEN	RET
RU6	6	R	CEN	RET	Ra6	6	R	CEN	RET
Ra:6	6	R	CEN	RET	Ri:6	6	R	CEN	RET
Ru:6	6	R	CEN	RET	h2:	2	h2	FRO	FRO
h9	9	h9	FRO	FRO	h9:	9	h9	FRO	FRO
ha	a	ha	CEN	CEN	hA	A	hA	CEN	CEN
ha:	a	ha	CEN	CEN	haI	aI	ha	FRO	CEN
haU	aU	ha	BAC	CEN	heI	eI	hE	FRO	FRO

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
hoU	oU	hoU	BAC	BAC	he:	e	he	FRO	FRO
hE	E	hE	FRO	FRO	hE	E	hE	FRO	FRO
hE:	E	hE	FRO	FRO	hI	I	hI	FRO	FRO
hi:	i	hi	FRO	FRO	ho:	o	ho	BAC	BAC
hO	O	hO	BAC	BAC	hO	O	hO	BAC	BAC
hO:	O	hO	BAC	BAC	hOY	OY	hO	FRO	BAC
hu:	u	hu	BAC	BAC	hU	U	hU	BAC	BAC
hY	Y	hY	FRO	FRO	hy:	y	hy	FRO	FRO
h2:6	6	h2	CEN	FRO	h96	6	h9	CEN	FRO
ha6	6	ha	CEN	CEN	ha:6	6	ha	CEN	CEN
he:6	6	he	CEN	FRO	hE6	6	hE	CEN	FRO
hE:6	6	hE	CEN	FRO	hI6	6	hI	CEN	FRO
hi:6	6	hi	CEN	FRO	ho:6	6	ho	CEN	BAC
hO6	6	hO	CEN	BAC	hO:6	6	hO	CEN	BAC
hu:6	6	hu	CEN	BAC	hU6	6	hU	CEN	BAC
hY6	6	hY	CEN	FRO	hy:6	6	hy	CEN	FRO
v2:	2	v	FRO	LAB	v9	9	v	FRO	LAB
v9:	9	v	FRO	LAB	v6	6	v	CEN	LAB
v@	@	v	CEN	LAB	va	a	v	CEN	LAB
vA	A	v	CEN	LAB	va:	a	v	CEN	LAB
vaI	aI	v	FRO	LAB	vaU	aU	v	BAC	LAB
veI	eI	v	FRO	LAB	voU	oU	v	BAC	LAB
ve:	e	v	FRO	LAB	vE	E	v	FRO	LAB
vE	E	v	FRO	LAB	vE:	E	v	FRO	LAB
vI	I	v	FRO	LAB	vi:	I	v	FRO	LAB
vo:	o	v	BAC	LAB	vO	O	v	BAC	LAB
vO	O	v	BAC	LAB	vO:	O	v	BAC	LAB
vOY	OY	v	FRO	LAB	vu:	u	v	BAC	LAB
vU	U	v	BAC	LAB	vY	Y	v	FRO	LAB

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
vy:	y	v	FRO	LAB	v2:6	6	v	CEN	LAB
v96	6	v	CEN	LAB	va6	6	v	CEN	LAB
va:6	6	v	CEN	LAB	ve:6	6	v	CEN	LAB
vE6	6	v	CEN	LAB	vE:6	6	v	CEN	LAB
vI6	6	v	CEN	LAB	vi:6	6	v	CEN	LAB
vo:6	6	v	CEN	LAB	vO6	6	v	CEN	LAB
vO:6	6	v	CEN	LAB	vu:6	6	v	CEN	LAB
vU6	6	v	CEN	LAB	vY6	6	v	CEN	LAB
vy:6	6	v	CEN	LAB	w6	6	w	CEN	BIL
w@	@	w	CEN	BIL	wa	a	w	CEN	BIL
wA	A	w	CEN	BIL	wa:	a	w	CEN	BIL
waI	aI	w	FRO	BIL	waU	aU	w	BAC	BIL
weI	eI	w	FRO	BIL	woU	ou	w	BAC	BIL
wE	E	w	FRO	BIL	wE	E	w	FRO	BIL
wE:	E	w	FRO	BIL	wI	I	w	FRO	BIL
wi:	i	w	FRO	BIL	wO	O	w	BAC	BIL
wO	O	w	BAC	BIL	wO:	O	w	BAC	BIL
wOY	OY	w	FRO	BIL	wu:	u	w	BAC	BIL
wU	U	w	BAC	BIL	w9	9	w	FRO	BIL
w9:	9	w	FRO	BIL	w96	6	w	CEN	BIL
wa6	6	w	CEN	BIL	wa:6	6	w	CEN	BIL
wE:6	6	w	CEN	BIL	wI6	6	w	CEN	BIL
wi:6	6	w	CEN	BIL	wO6	6	w	CEN	BIL
wO:6	6	w	CEN	BIL	wu:6	6	w	CEN	BIL
wU6	6	w	CEN	BIL	r	r	r	UVU	UVU
$p	$p	$p	$p	$p	$j	$j	$j	$j	$j
§A	$p	$p	$p	$p	§B	$p	$p	$p	$p
§C	$p	$p	$p	$p	§D	$p	$p	$p	$p
§E	$p	$p	$p	$p	§F	$p	$p	$p	$p

Fortsetzung auf der nächsten Seite ...

Tabelle A.2.1.: Phoxsy-Einheiten mit ihren Kontextklassen
(Fortsetzung)

TKey	CCLeft	CCRight	CCLeft2	CCRight2	TKey	CCLeft	CCRight	CCLeft2	CCRight2
§G	$p	$p	$p	$p	§H	$p	$p	$p	$p
§I	$p	$p	$p	$p	§J	$p	$p	$p	$p
§K	$p	$p	$p	$p	§L	$p	$p	$p	$p
§M	$p	$p	$p	$p	§N	$p	$p	$p	$p
§O	$p	$p	$p	$p	§P	$p	$p	$p	$p
§Q	$p	$p	$p	$p	§R	$p	$p	$p	$p
§S	$p	$p	$p	$p	§T	$p	$p	$p	$p
§U	$p	$p	$p	$p	§V	$p	$p	$p	$p
§W	$p	$p	$p	$p	§X	$p	$p	$p	$p
§Y	$p	$p	$p	$p	§Z	$p	$p	$p	$p
§Ä	$p	$p	$p	$p	§Ö	$p	$p	$p	$p
§Ü	$p	$p	$p	$p	§ß	$p	$p	$p	$p
§0	$p	$p	$p	$p	§1	$p	$p	$p	$p
§2	$p	$p	$p	$p	§3	$p	$p	$p	$p
§4	$p	$p	$p	$p	§5	$p	$p	$p	$p
§6	$p	$p	$p	$p	§7	$p	$p	$p	$p
§8	$p	$p	$p	$p	§9	$p	$p	$p	$p

Anhang B. BOSS-Konfigurationsoptionen

B.1. Auswirkung der Schaltereinstellungen

Type	Flag	Value	Resultat
1	1 oder 0	gesetzt	Option ist vom Argument-Typ, Schalterwert ist ,,ein”
1	1 oder 0	leer	Option ist vom Argument-Typ, Programmabbruch
0	0	leer oder gesetzt	Option ist vom Binär-Typ, Schalterwert ist ,,aus”
0	1	leer oder gesetzt	Option ist vom Binär-Typ, Schalterwert ist ,,ein”

Tabelle B.1.1.: Verarbeitung verschiedener Attributwerte in BOSS-Konfigurationsdateien durch die Klasse `BOSS::Config` (vgl. Abschnitt 5.1.1).

B.2. Auszug aus einer Konfigurationsdatei

```
1 <?xml version = '1.0'?>
2 <CONFIGS DefaultInventory="lioba_b">
3   <INVENTORY Name="lioba_b" Lang="de" >
4     <OPTION Comment="The␣module␣order␣definition␣file." Name="modules" Value
          ="/usr/local/share/boss3/data/de/lioba/runtime/boss_sampa/
          module_order.txt" Type="1" Flag="1" />
5     <OPTION Comment="The␣language␣suffix␣of␣the␣cost␣function␣library␣to␣be␣
          used␣in␣unit␣selection" Name="cf" Value="de" Type="1" Flag="1" />
6     <OPTION Comment="The␣symbol␣table␣for␣morpheme␣decomposition." Name="
          morphsym" Value="/usr/local/share/boss3/data/de/lioba/runtime/
          boss_sampa/phones.txt" Type="1" Flag="1" />
7 ...
```

Listing B.1: Beginn der Konfigurationsdatei für die BOSS-SAMPA-Version der deutschen Lioba-Stimme (ohne phoxsy-Label) — vgl. Abschnitt 5.1.1.

Anhang C. XML-Schemata

C.1. Inventar-XML-Schema

```
1  <?xml version="1.0" ?>
2  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3
4          <xsd:annotation>
5                  <xsd:documentation>
6                  XML Schema for BOSS (DE) inventory annotation
7                  documents (derived from BLF label files).
8                  This version is for German BOSS systems
9                  using the classes provided with the BOSS_DE
10                 distributions.
11                 $Id$
12                 Copyright (C) 2003 University of Bonn (IKP).
13                 Author: Stefan Breuer
14                 </xsd:documentation>
15         </xsd:annotation>
16
17         <!-- Type definition of the values of the sentence-level "Type"
               attribute.
18         This attribute is set by the tool blf2xml depending on the
               annotation in the
19         source BLF file -->
20         <xsd:simpleType name="SentenceTypes">
21                 <xsd:restriction base="xsd:string">
22                         <!-- Falling pitch -->
23                         <xsd:enumeration value="." />
24                         <!-- Rising pitch -->
25                         <xsd:enumeration value="?" />
26                 </xsd:restriction>
27         </xsd:simpleType>
28
29         <!-- Type definition of the values of the "PMode" attribute.
30         This attribute is set at word level by the tool blf2xml and
31         at syllable and phone level by the tool addphrases -->
32         <xsd:simpleType name="PhraseBoundTypes">
33                 <xsd:restriction base="xsd:string">
34                         <!-- Falling pitch -->
```

```
35                      <xsd:enumeration value="." />
36                      <!-- Rising pitch -->
37                      <xsd:enumeration value="?" />
38                      <!-- No phrase boundary present (medial position) -->
39                      <xsd:enumeration value="" />
40              </xsd:restriction>
41      </xsd:simpleType>
42
43      <!-- Type definition of the values of the "Align" attribute.
44      This attribute is set by the tool alignflag -->
45      <xsd:simpleType name="AlignTypes">
46              <xsd:restriction base="xsd:string">
47                      <!-- Automatic segmentation (i.e. not used for
                            synthesis) -->
48                      <xsd:enumeration value="Auto" />
49                      <!-- Manual segmentation -->
50                      <xsd:enumeration value="Manual" />
51              </xsd:restriction>
52      </xsd:simpleType>
53
54      <!-- Type definition of the values of the "Half" attribute for half-
            phones -->
55      <xsd:simpleType name="HalfPhoneSegments">
56              <xsd:restriction base="xsd:positiveInteger">
57                      <!-- First half (initial segment) -->
58                      <xsd:enumeration value="1" />
59                      <!-- Second half (final segment) -->
60                      <xsd:enumeration value="2" />
61              </xsd:restriction>
62      </xsd:simpleType>
63
64      <!-- Type definition of the values of the "Stress" attribute -->
65      <xsd:simpleType name="StressTypes">
66              <xsd:restriction base="xsd:nonNegativeInteger">
67                      <!-- No lexical stress -->
68                      <xsd:enumeration value="0" />
69                      <!-- Primary lexical stress -->
70                      <xsd:enumeration value="1" />
71                      <!-- Secondary lexical stress -->
72                      <xsd:enumeration value="2" />
73              </xsd:restriction>
74      </xsd:simpleType>
75
76      <!-- Description of a Sentence element
77  -->
78      <xsd:element name="SENTENCE">
79              <xsd:complexType>
```

```
80                      <xsd:sequence>
81                              <xsd:element ref="WORD" minOccurs="1" maxOccurs
                                    ="unbounded" />
82                      </xsd:sequence>
83                      <!-- the sentence type (question or statement,
84                      see SentenceTypes definition) -->
85                      <xsd:attribute name="Type" type="SentenceTypes" use="
                            required" />
86                      <!-- the base name of the XML file -->
87                      <xsd:attribute name="File" type="xsd:string" />
88              </xsd:complexType>
89      </xsd:element>
90
91      <!--   Type definition of attributes common to all elements below
            sentence level.
92  -->
93      <xsd:attributeGroup name="CommonAttrWSPH">
94              <!-- the canonical transcription of the element
95               (without stress or boundary markers) -->
96              <xsd:attribute name="TKey" type="xsd:string" />
97      <!-- the transcription of the actual pronunciation
98       of the element (without stress or boundary markers)
99       Added by blf2xml. -->
100     <xsd:attribute name="TReal" type="xsd:string" />
101             <!-- the intensity and place of a phrase boundary marker
102              (-5 to 5, where -5 is initial and 5 is final)
103               Added by blf2xml. -->
104             <xsd:attribute name="PInt" type="xsd:int" />
105             <!-- the type of phrase boundary
106             (see the definition of "PhraseBoundTypes" above)
107             Added by blf2xml. -->
108             <xsd:attribute name="PMode" type="PhraseBoundTypes" />
109             <!-- the left phone context of the element -
110             used by the pre-selection algorithm
111             Added by blf2xml. -->
112             <xsd:attribute name="CLeft" type="xsd:string" />
113             <!-- the right phone context of the element
114             Added by blf2xml. -->
115             <xsd:attribute name="CRight" type="xsd:string" />
116             <!-- the phone before the left neighbouring phone
117              of the element.
118              Added by blf2xml. -->
119             <xsd:attribute name="CLeft2" type="xsd:string" />
120             <!-- the phone after the right neighbouring phone
121              of the element.
122              Added by blf2xml. -->
123             <xsd:attribute name="CRight2" type="xsd:string" />
```

```
124             <!-- Each context phone in BOSS is assigned to two
                    classification
125              schemas, which can be defined by the user.
126             The default classification groups all symbols that are
                    closely
127             related or even identical in the relevant sections
128              on the first level (e.g. [ja:] and [la:] are class [a:]
129              as left context; note that BOSS_DE uses multi-phone units),
130             and assigns all sounds to the same class that share the place
131              of articulation on the second level. All context class
                     attributes
132               are provided by the tool blf2xml.
133              The following attribute lists which
134              (first level) class the left context (multi-)phone
135              belongs to -->
136             <xsd:attribute name="CCLeft" type="xsd:string" />
137             <!-- The first level context class of the right
138              neighbour of the element -->
139             <xsd:attribute name="CCRight" type="xsd:string" />
140             <!-- The second level context class of the left
141              neighbour of the element -->
142             <xsd:attribute name="CCLeft2" type="xsd:string" />
143             <!-- The second level context class of the right
144              neighbour of the element -->
145             <xsd:attribute name="CCRight2" type="xsd:string" />
146             <!-- The position of the element in the source signal file
147              (left boundary, in samples). Added by blf2xml. -->
148             <xsd:attribute name="First" type="xsd:nonNegativeInteger" />
149             <!-- The position of the element in the source signal file
150              (right boundary, in samples). Added by blf2xml. -->
151         <xsd:attribute name="Last" type="xsd:nonNegativeInteger" />
152             <!-- F0 value at the left boundary of the element (avg) -->
153             <xsd:attribute name="LF0" type="xsd:double" />
154             <!-- F0 value at the right boundary of the element (avg).
155             Provided by addf0. -->
156         <xsd:attribute name="RF0" type="xsd:double" />
157         <!-- the average F0 value of the whole element.
158          Provided by addf0. -->
159         <xsd:attribute name="AVGF0" type="xsd:double" />
160         <!-- average RMS value of the whole element.
161         Provided by addf0. -->
162         <xsd:attribute name="RMS" type="xsd:double" />
163         <!-- Mel Frequency Cepstrum Coefficients (LM0-10)
164         at the left element boundary. Provided by melbounds. -->
165         <xsd:attribute name="LM0" type="xsd:double" />
166         <xsd:attribute name="LM1" type="xsd:double" />
167         <xsd:attribute name="LM2" type="xsd:double" />
```

```
168         <xsd:attribute name="LM3" type="xsd:double" />
169         <xsd:attribute name="LM4" type="xsd:double" />
170         <xsd:attribute name="LM5" type="xsd:double" />
171         <xsd:attribute name="LM6" type="xsd:double" />
172         <xsd:attribute name="LM7" type="xsd:double" />
173         <xsd:attribute name="LM8" type="xsd:double" />
174         <xsd:attribute name="LM9" type="xsd:double" />
175         <xsd:attribute name="LM10" type="xsd:double" />
176         <!-- Mel Frequency Cepstrum Coefficients (RM0-10)
177         at the right element boundary. Provided by melbounds. -->
178         <xsd:attribute name="RM0" type="xsd:double" />
179         <xsd:attribute name="RM1" type="xsd:double" />
180         <xsd:attribute name="RM2" type="xsd:double" />
181         <xsd:attribute name="RM3" type="xsd:double" />
182         <xsd:attribute name="RM4" type="xsd:double" />
183         <xsd:attribute name="RM5" type="xsd:double" />
184         <xsd:attribute name="RM6" type="xsd:double" />
185         <xsd:attribute name="RM7" type="xsd:double" />
186         <xsd:attribute name="RM8" type="xsd:double" />
187         <xsd:attribute name="RM9" type="xsd:double" />
188         <xsd:attribute name="RM10" type="xsd:double" />
189                 <!-- An attribute to mark elements as either manually or
190                  automatically aligned. Can also be used for pruning or
                        temporary
191                  removal of elements from the corpus. Provided by the tool
                        alignflag. -->
192         <xsd:attribute name="Align" type="AlignTypes" />
193         <!-- Fujisaki parameters for the description of F0 contours.
194          Provided by addfuji. -->
195         <xsd:attribute name="Aa" type="xsd:double" />
196         <xsd:attribute name="Ap" type="xsd:double" />
197         <xsd:attribute name="T0" type="xsd:double" />
198         <xsd:attribute name="T1" type="xsd:double" />
199         <xsd:attribute name="T2" type="xsd:double" />
200         </xsd:attributeGroup>
201
202         <!-- Description of a Word element -->
203         <xsd:element name="WORD">
204                 <xsd:complexType>
205                         <xsd:sequence>
206                                 <xsd:element ref="SYLLABLE" minOccurs="0"
                                        maxOccurs="unbounded" />
207                         </xsd:sequence>
208                         <xsd:attributeGroup ref="CommonAttrWSPH" />
209                 </xsd:complexType>
210         </xsd:element>
211
```

```
212         <!-- Description of a Syllable element -->
213         <xsd:element name="SYLLABLE">
214                 <xsd:complexType>
215                         <xsd:sequence>
216                                 <xsd:element ref="PHONE" minOccurs="0"
                                        maxOccurs="unbounded" />
217                         </xsd:sequence>
218                         <xsd:attributeGroup ref="CommonAttrWSPH" />
219                         <!-- the presence and type of lexical stress is marked
                                by
220                          this attribute (see the definition of "StressTypes"
                                above) -->
221                         <xsd:attribute name="Stress" type="StressTypes" />
222                 </xsd:complexType>
223         </xsd:element>
224
225         <!-- Description of a Phone element -->
226         <xsd:element name="PHONE">
227                 <xsd:complexType>
228                         <xsd:sequence>
229                                 <xsd:element ref="HALFPHONE" minOccurs="0"
                                        maxOccurs="2" />
230                         </xsd:sequence>
231                         <xsd:attributeGroup ref="CommonAttrWSPH" />
232                         <xsd:attribute name="Stress" type="StressTypes" />
233                 </xsd:complexType>
234         </xsd:element>
235
236         <!-- Description of a Half-Phone element -->
237         <xsd:element name="HALFPHONE">
238                 <xsd:complexType>
239                         <xsd:attributeGroup ref="CommonAttrWSPH" />
240                         <!-- This attribute marks if the element is an
241                          inital or final half-phone unit -->
242                         <xsd:attribute name="Half" type="HalfPhoneSegments" />
243                         <xsd:attribute name="Stress" type="StressTypes" />
244                 </xsd:complexType>
245         </xsd:element>
246 </xsd:schema>
```

Listing C.1: Schema für deutsche Inventar-XML-Dokumente (vgl. Abschnitt 4.1.2).

C.2. Server-XML-Schema

```
1  <?xml version="1.0" ?>
2  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3
4          <xsd:annotation>
5                  <xsd:documentation>
6                  XML Schema for BOSS (DE) server documents.
7                  This version is for German BOSS systems
8                  using the classes provided with the BOSS_DE
9                  distributions.
10                 $Id$
11                 Copyright (C) 2003 University of Bonn (IKP).
12                 Authors: Tim Schroeder, Stefan Breuer
13                 </xsd:documentation>
14         </xsd:annotation>
15
16         <!-- Type definition for the values of the "Phrpos" attribute.
17         This attribute is set and used by the class boss_duration_de -->
18         <xsd:simpleType name="PosInPhrase">
19                 <xsd:restriction base="xsd:string">
20                         <!-- Initial position -->
21                         <xsd:enumeration value="I" />
22                         <!-- Medial position -->
23                         <xsd:enumeration value="M" />
24                         <!-- Final position -->
25                         <xsd:enumeration value="F" />
26                 </xsd:restriction>
27         </xsd:simpleType>
28
29         <!-- Type definition for the values of the sentence-level "Type"
               attribute.
30         This attribute is set by the client -->
31         <xsd:simpleType name="SentenceTypes">
32                 <xsd:restriction base="xsd:string">
33                         <!-- Falling pitch -->
34                         <xsd:enumeration value="." />
35                         <!-- Rising pitch -->
36                         <xsd:enumeration value="?" />
37                 </xsd:restriction>
38         </xsd:simpleType>
39
40         <!-- Type definition for the values of the "PMode" attribute.
41         This attribute is set by the class boss_transcription_de -->
42         <xsd:simpleType name="PhraseBoundTypes">
43                 <xsd:restriction base="xsd:string">
44                         <!-- Falling pitch -->
```

```
45                      <xsd:enumeration value="." />
46                      <!-- Rising pitch -->
47                      <xsd:enumeration value="?" />
48                      <!-- No phrase boundary present (medial position) -->
49                      <xsd:enumeration value="" />
50              </xsd:restriction>
51      </xsd:simpleType>
52
53      <!-- Type definition for the values of the "Half" attribute for half
            -phones -->
54      <xsd:simpleType name="HalfPhoneSegments">
55              <xsd:restriction base="xsd:positiveInteger">
56                      <!-- First half (initial segment) -->
57                      <xsd:enumeration value="1" />
58                      <!-- Second half (final segment) -->
59                      <xsd:enumeration value="2" />
60              </xsd:restriction>
61      </xsd:simpleType>
62
63      <!-- Type definition for the values of the "Stress" attribute -->
64      <xsd:simpleType name="StressTypes">
65              <xsd:restriction base="xsd:nonNegativeInteger">
66                      <!-- No lexical stress -->
67                      <xsd:enumeration value="0" />
68                      <!-- Primary lexical stress -->
69                      <xsd:enumeration value="1" />
70                      <!-- Secondary lexical stress -->
71                      <xsd:enumeration value="2" />
72              </xsd:restriction>
73      </xsd:simpleType>
74
75      <!--
76   Text Node.
77   -->
78      <xsd:element name="TEXT">
79              <xsd:complexType>
80                      <xsd:sequence>
81                              <xsd:element ref="SENTENCE" minOccurs="1"
                                    maxOccurs="1" />
82                      </xsd:sequence>
83              </xsd:complexType>
84      </xsd:element>
85
86      <!-- Description of a Sentence
87  -->
88      <xsd:element name="SENTENCE">
89              <xsd:complexType>
```

```
90                      <xsd:sequence>
91                              <xsd:element ref="WORD" minOccurs="1" maxOccurs
                                    ="unbounded" />
92                      </xsd:sequence>
93                      <!-- the sentence type (question or statement,
94                      see SentenceTypes definition) -->
95                      <xsd:attribute name="Type" type="SentenceTypes" use="
                            required" />
96                      <!-- an optional ID for the sentence, used in file
                            naming -->
97                      <xsd:attribute name="SentenceID" type="xsd:string" use
                            ="optional" />
98                      <!-- an optional path for the output signal file
99                       (if using file mode) -->
100                     <xsd:attribute name="TargetPath" type="xsd:string" use
                            ="optional" />
101                     <!-- an optional name (file mask) for the output
                            signal file
102                      (if using file mode) -->
103                     <xsd:attribute name="TargetFile" type="xsd:string" use
                            ="optional" />
104             </xsd:complexType>
105     </xsd:element>
106
107     <!--    Type definition for attributes common to all elements below
                sentence level.
108 -->
109     <xsd:attributeGroup name="CommonAttrWSPH">
110             <!-- The position of the element in the phrase (either I,M,
                    or F) -->
111             <xsd:attribute name="Phrpos" type="PosInPhrase" />
112             <!-- The duration of the element in ms -->
113             <xsd:attribute name="Dur" type="xsd:nonNegativeInteger" />
114             <!-- the transcription of the element (without stress or
                    boundary markers) -->
115             <xsd:attribute name="TKey" type="xsd:string" />
116             <!-- the intensity and place of a phrase boundary marker
117              (-5 to 5, where -5 is initial and 5 is final) -->
118             <xsd:attribute name="PInt" type="xsd:int" />
119             <!-- the type of phrase boundary
120             (see the definition of "PhraseBoundTypes" above) -->
121             <xsd:attribute name="PMode" type="PhraseBoundTypes" />
122             <!-- the left phone context of the element -
123             used by the pre-selection algorithm -->
124             <xsd:attribute name="CLeft" type="xsd:string" />
125             <!-- the right phone context of the element -->
126             <xsd:attribute name="CRight" type="xsd:string" />
```

```
127             <!-- the phone before the left neighbouring phone
128              of the element. -->
129             <xsd:attribute name="CLeft2" type="xsd:string" />
130             <!-- the phone after the right neighbouring phone
131              of the element. -->
132             <xsd:attribute name="CRight2" type="xsd:string" />
133             <!-- Each context phone in BOSS is assigned to two
                    classification
134              schemas, which can be defined by the user.
135             The default classification groups all symbols that are
                    closely
136             related or even identical in the relevant sections
137              on the first level (e.g. [ja:] and [la:] are class [a:]
138              as left context; note that BOSS_DE uses multi-phone units),
139             and assigns all sounds to the same class that share the place
140              of articulation on the second level. This is needed during
141              pre-selection, if the desired phone is not available in the
142              exact phonetic context. The following attribute lists which
143              (first level) class the left context (multi-)phone
144              belongs to -->
145             <xsd:attribute name="CCLeft" type="xsd:string" />
146             <!-- The first level context class of the right
147              neighbour of the element -->
148             <xsd:attribute name="CCRight" type="xsd:string" />
149             <!-- The second level context class of the left
150              neighbour of the element -->
151             <xsd:attribute name="CCLeft2" type="xsd:string" />
152             <!-- The second level context class of the right
153              neighbour of the element -->
154             <xsd:attribute name="CCRight2" type="xsd:string" />
155     </xsd:attributeGroup>
156
157     <!-- Description of a Word -->
158     <xsd:element name="WORD">
159             <xsd:complexType>
160                     <xsd:sequence>
161                             <xsd:element ref="SYLLABLE" minOccurs="0"
                                    maxOccurs="unbounded" />
162                     </xsd:sequence>
163                     <xsd:attributeGroup ref="CommonAttrWSPH" />
164                     <!-- The orthographic string -->
165                     <xsd:attribute name="Orth" type="xsd:string" use="
                            required" />
166                     <!-- an attribute for pre-defined transcriptions -->
167                     <xsd:attribute name="WordTrans" type="xsd:string" />
168                     <!-- an index for the phonemes within an element -->
```

```
169                     <xsd:attribute name="PNr" type="xsd:nonNegativeInteger
                            " />
170             </xsd:complexType>
171         </xsd:element>
172
173         <!-- Description of a Syllable -->
174         <xsd:element name="SYLLABLE">
175                 <xsd:complexType>
176                         <xsd:sequence>
177                                 <xsd:element ref="PHONE" minOccurs="0"
                                        maxOccurs="unbounded" />
178                         </xsd:sequence>
179                         <xsd:attributeGroup ref="CommonAttrWSPH" />
180                         <!-- The number of syllables between this syllable and
                                the nearest
181                          phrase boundary (as defined for BPD, the BONN
                                Prosodic Database) -->
182                         <xsd:attribute name="Bodi" type="
                                xsd:nonNegativeInteger" />
183                         <xsd:attribute name="PNr" type="xsd:nonNegativeInteger
                                " />
184                         <!-- the presence and type of lexical stress is marked
                                by
185                          this attribute (see the definition of "StressTypes"
                                above) -->
186                         <xsd:attribute name="Stress" type="StressTypes" />
187                 </xsd:complexType>
188         </xsd:element>
189
190         <!-- Description of a Phone -->
191         <xsd:element name="PHONE">
192                 <xsd:complexType>
193                         <xsd:sequence>
194                                 <xsd:element ref="HALFPHONE" minOccurs="0"
                                        maxOccurs="2" />
195                         </xsd:sequence>
196                         <xsd:attributeGroup ref="CommonAttrWSPH" />
197                         <xsd:attribute name="Stress" type="StressTypes" />
198                 </xsd:complexType>
199         </xsd:element>
200
201         <!-- Description of a Half-Phone -->
202         <xsd:element name="HALFPHONE">
203                 <xsd:complexType>
204                         <xsd:attributeGroup ref="CommonAttrWSPH" />
205                         <!-- This attribute marks if the element is an
206                          inital or final half-phone unit -->
```

```
207                         <xsd:attribute name="Half" type="HalfPhoneSegments" />
208                         <xsd:attribute name="Stress" type="StressTypes" />
209                 </xsd:complexType>
210         </xsd:element>
211 </xsd:schema>
```

Listing C.2: Schema für deutsche Server-XML-Dokumente (vgl. Abschnitt 4.2.1).

Anhang D. Erweiterte SQL-Schemadefinition für BOSS-Datenbanken

```
1  CREATE DATABASE /*!32312 IF NOT EXISTS*/ `boss_de_lioba_p` /*!40100 DEFAULT
       CHARACTER SET latin1 */;
2
3  USE `boss_de_lioba_p`;
4
5  --
6  -- Table structure for table `exception_lexicon`
7  --
8
9  DROP TABLE IF EXISTS `exception_lexicon`;
10 CREATE TABLE `exception_lexicon` (
11   `Orth` varchar(50) character set latin1 collate latin1_bin NOT NULL
         default '',
12   `Trans` varchar(50) character set latin1 collate latin1_bin NOT NULL
         default '',
13   PRIMARY KEY (`Orth`)
14 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
15
16 --
17 -- Table structure for table `lioba_p_halfphone_data`
18 --
19
20 DROP TABLE IF EXISTS `lioba_p_halfphone_data`;
21 CREATE TABLE `lioba_p_halfphone_data` (
22   `Num` int(10) unsigned NOT NULL default '0',
23   `TKey` char(5) character set latin1 collate latin1_bin NOT NULL default ''
         ,
24   `TReal` char(12) character set latin1 collate latin1_bin NOT NULL default
         '',
25   `Half` int(10) unsigned NOT NULL default '0',
26   `CLeft` char(5) character set latin1 collate latin1_bin NOT NULL default '
         ',
27   `CRight` char(5) character set latin1 collate latin1_bin NOT NULL default
         '',
```

```
28    `CCLeft` char(5) character set latin1 collate latin1_bin NOT NULL default
          '',
29    `CCRight` char(5) character set latin1 collate latin1_bin NOT NULL default
          '',
30    `PInt` int(10) unsigned NOT NULL default '0',
31    `PMode` enum('.','?') NOT NULL default '.',
32    `First` int(10) unsigned NOT NULL default '0',
33    `Last` int(10) unsigned NOT NULL default '0',
34    `Stress` enum('0','1','2') NOT NULL default '0',
35    `Align` enum('Manual','Auto') NOT NULL default 'Manual',
36    `LM0` float default NULL,
37    `LM1` float default NULL,
38    `LM2` float default NULL,
39    `LM3` float default NULL,
40    `LM4` float default NULL,
41    `LM5` float default NULL,
42    `LM6` float default NULL,
43    `LM7` float default NULL,
44    `LM8` float default NULL,
45    `LM9` float default NULL,
46    `LM10` float default NULL,
47    `RM0` float default NULL,
48    `RM1` float default NULL,
49    `RM2` float default NULL,
50    `RM3` float default NULL,
51    `RM4` float default NULL,
52    `RM5` float default NULL,
53    `RM6` float default NULL,
54    `RM7` float default NULL,
55    `RM8` float default NULL,
56    `RM9` float default NULL,
57    `RM10` float default NULL,
58    `AVGF0` float default NULL,
59    `LF0` float default NULL,
60    `RF0` float default NULL,
61    `RMS` float default NULL,
62    `T0` float default NULL,
63    `Ap` float default NULL,
64    `T1` float default NULL,
65    `T2` float default NULL,
66    `Aa` float default NULL,
67    PRIMARY KEY (`Num`),
68    KEY `TKey` (`TKey`,`Align`,`Half`,`CLeft`,`CRight`,`PInt`,`Stress`),
69    KEY `TKey_2` (`TKey`,`Align`,`Half`,`CCLeft`,`CRight`,`PInt`,`Stress`),
70    KEY `TKey_3` (`TKey`,`Align`,`Half`,`CLeft`,`CCRight`,`PInt`,`Stress`),
71    KEY `TKey_4` (`TKey`,`Align`,`Half`,`CCLeft`,`CCRight`,`PInt`,`Stress`),
72    KEY `TKey_5` (`TKey`,`Align`,`Half`,`PInt`,`CLeft`,`Stress`),
```

```
  KEY `TKey_6` (`TKey`,`Align`,`Half`,`Stress`,`CLeft`),
  KEY `TKey_7` (`TKey`,`Align`,`Half`,`CRight`,`PInt`,`Stress`),
  KEY `TKey_8` (`TKey`,`Align`,`Half`,`CRight`,`Stress`),
  KEY `TKey_9` (`TKey`,`Align`,`Half`,`CCRight`,`PInt`,`Stress`),
  KEY `TKey_10` (`TKey`,`Align`,`Half`,`CCRight`,`Stress`),
  KEY `TKey_11` (`TKey`,`Align`,`Half`,`CCLeft`,`PInt`,`Stress`),
  KEY `TKey_12` (`TKey`,`Align`,`Half`,`CCLeft`,`Stress`),
  KEY `TKey_13` (`TKey`,`Align`,`Half`,`PInt`,`Stress`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Table structure for table `lioba_p_phone_data`
--

DROP TABLE IF EXISTS `lioba_p_phone_data`;
CREATE TABLE `lioba_p_phone_data` (
  `Num` int(10) unsigned NOT NULL default '0',
  `TKey` varchar(5) character set latin1 collate latin1_bin NOT NULL default
      '',
  `TReal` varchar(12) character set latin1 collate latin1_bin NOT NULL
     default '',
  `CLeft` varchar(5) character set latin1 collate latin1_bin NOT NULL
     default '',
  `CRight` varchar(5) character set latin1 collate latin1_bin NOT NULL
     default '',
  `CCLeft` varchar(5) character set latin1 collate latin1_bin NOT NULL
     default '',
  `CCRight` varchar(5) character set latin1 collate latin1_bin NOT NULL
     default '',
  `PInt` int(10) unsigned NOT NULL default '0',
  `PMode` enum('.','?') NOT NULL default '.',
  `First` int(10) unsigned NOT NULL default '0',
  `Last` int(10) unsigned NOT NULL default '0',
  `Stress` enum('0','1','2') NOT NULL default '0',
  `Align` enum('Manual','Auto') NOT NULL default 'Manual',
  `LM0` float default NULL,
  `LM1` float default NULL,
  `LM2` float default NULL,
  `LM3` float default NULL,
  `LM4` float default NULL,
  `LM5` float default NULL,
  `LM6` float default NULL,
  `LM7` float default NULL,
  `LM8` float default NULL,
  `LM9` float default NULL,
  `LM10` float default NULL,
  `RM0` float default NULL,
```

```
114   `RM1` float default NULL,
115   `RM2` float default NULL,
116   `RM3` float default NULL,
117   `RM4` float default NULL,
118   `RM5` float default NULL,
119   `RM6` float default NULL,
120   `RM7` float default NULL,
121   `RM8` float default NULL,
122   `RM9` float default NULL,
123   `RM10` float default NULL,
124   `AVGF0` float default NULL,
125   `LF0` float default NULL,
126   `RF0` float default NULL,
127   `RMS` float default NULL,
128   `CCLeft2` varchar(100) NOT NULL default '',
129   `CCRight2` varchar(100) NOT NULL default '',
130   `T0` float default NULL,
131   `Ap` float default NULL,
132   `T1` float default NULL,
133   `T2` float default NULL,
134   `Aa` float default NULL,
135   PRIMARY KEY (`Num`),
136   KEY `TKey_2` (`TKey`,`CCLeft`,`CRight`,`Align`,`PInt`,`Stress`),
137   KEY `TKey_3` (`TKey`,`CLeft`,`CCRight`,`Align`,`PInt`,`Stress`),
138   KEY `TKey_4` (`TKey`,`CCLeft`,`CCRight`,`Align`,`PInt`,`Stress`),
139   KEY `TKey_5` (`TKey`,`Align`,`PInt`,`CLeft`,`Stress`),
140   KEY `TKey_10` (`TKey`,`Align`,`CCRight`,`Stress`),
141   KEY `TKey_6` (`TKey`,`Align`,`Stress`,`CLeft`),
142   KEY `TKey_7` (`TKey`,`Align`,`CRight`,`PInt`,`Stress`),
143   KEY `TKey_11` (`TKey`,`Align`,`CCLeft`,`PInt`,`Stress`),
144   KEY `TKey` (`TKey`,`CLeft`,`CRight`,`Align`,`PInt`,`Stress`),
145   KEY `TKey_8` (`TKey`,`Align`,`CRight`,`Stress`),
146   KEY `TKey_12` (`TKey`,`Align`,`CCLeft`,`Stress`),
147   KEY `TKey_13` (`TKey`,`Align`,`PInt`,`Stress`),
148   KEY `TKey_9` (`TKey`,`Align`,`CCRight`,`PInt`,`Stress`)
149 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
150
151 --
152 -- Table structure for table `lioba_p_phone_map`
153 --
154
155 DROP TABLE IF EXISTS `lioba_p_phone_map`;
156 CREATE TABLE `lioba_p_phone_map` (
157   `PhoneNum` int(10) unsigned NOT NULL default '0',
158   `HalfphoneNum` int(10) unsigned NOT NULL default '0',
159   PRIMARY KEY (`HalfphoneNum`),
160   KEY `HalfphoneNum` (`HalfphoneNum`)
```

```
161 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
162
163 --
164 -- Table structure for table `lioba_p_sentence_data`
165 --
166
167 DROP TABLE IF EXISTS `lioba_p_sentence_data`;
168 CREATE TABLE `lioba_p_sentence_data` (
169   `Num` int(10) unsigned NOT NULL default '0',
170   `File` varchar(100) character set latin1 collate latin1_bin NOT NULL
        default '',
171   `Type` char(1) character set latin1 collate latin1_bin NOT NULL default ''
        ,
172   PRIMARY KEY (`Num`)
173 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
174
175 --
176 -- Table structure for table `lioba_p_sentence_map`
177 --
178
179 DROP TABLE IF EXISTS `lioba_p_sentence_map`;
180 CREATE TABLE `lioba_p_sentence_map` (
181   `SentenceNum` int(10) unsigned NOT NULL default '0',
182   `WordNum` int(10) unsigned NOT NULL default '0',
183   PRIMARY KEY (`WordNum`),
184   KEY `SentenceNum` (`SentenceNum`),
185   KEY `WordNum` (`WordNum`)
186 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
187
188 --
189 -- Table structure for table `lioba_p_syllable_data`
190 --
191
192 DROP TABLE IF EXISTS `lioba_p_syllable_data`;
193 CREATE TABLE `lioba_p_syllable_data` (
194   `Num` int(10) unsigned NOT NULL default '0',
195   `TKey` varchar(20) character set latin1 collate latin1_bin NOT NULL
        default '',
196   `TReal` varchar(40) character set latin1 collate latin1_bin NOT NULL
        default '',
197   `CLeft` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
198   `CRight` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
199   `CCLeft` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
```

```
200   `CCRight` varchar(5) character set latin1 collate latin1_bin NOT NULL
          default '',
201   `PInt` int(10) unsigned NOT NULL default '0',
202   `PMode` enum('.','?') NOT NULL default '.',
203   `First` int(10) unsigned NOT NULL default '0',
204   `Last` int(10) unsigned NOT NULL default '0',
205   `Stress` enum('0','1','2') NOT NULL default '0',
206   `Align` enum('Manual','Auto') NOT NULL default 'Manual',
207   `LM0` float default NULL,
208   `LM1` float default NULL,
209   `LM2` float default NULL,
210   `LM3` float default NULL,
211   `LM4` float default NULL,
212   `LM5` float default NULL,
213   `LM6` float default NULL,
214   `LM7` float default NULL,
215   `LM8` float default NULL,
216   `LM9` float default NULL,
217   `LM10` float default NULL,
218   `RM0` float default NULL,
219   `RM1` float default NULL,
220   `RM2` float default NULL,
221   `RM3` float default NULL,
222   `RM4` float default NULL,
223   `RM5` float default NULL,
224   `RM6` float default NULL,
225   `RM7` float default NULL,
226   `RM8` float default NULL,
227   `RM9` float default NULL,
228   `RM10` float default NULL,
229   `AVGF0` float default NULL,
230   `LF0` float default NULL,
231   `RF0` float default NULL,
232   `RMS` float default NULL,
233   `CCLeft2` varchar(100) NOT NULL default '',
234   `CCRight2` varchar(100) NOT NULL default '',
235   `T0` float default NULL,
236   `Ap` float default NULL,
237   `T1` float default NULL,
238   `T2` float default NULL,
239   `Aa` float default NULL,
240   PRIMARY KEY (`Num`),
241   KEY `TKey_2` (`TKey`,`Align`,`CCLeft`,`CRight`,`PInt`,`Stress`),
242   KEY `TKey_3` (`TKey`,`Align`,`CLeft`,`CCRight`,`PInt`,`Stress`),
243   KEY `TKey_4` (`TKey`,`Align`,`CCLeft`,`CCRight`,`PInt`,`Stress`),
244   KEY `TKey_5` (`TKey`,`Align`,`PInt`,`CLeft`,`Stress`),
245   KEY `TKey_10` (`TKey`,`Align`,`CCRight`,`Stress`),
```

```
246   KEY `TKey_6` (`TKey`,`Align`,`Stress`,`CLeft`),
247   KEY `TKey_7` (`TKey`,`Align`,`CRight`,`PInt`,`Stress`),
248   KEY `TKey_11` (`TKey`,`Align`,`CCLeft`,`PInt`,`Stress`),
249   KEY `TKey` (`TKey`,`Align`,`CLeft`,`CRight`,`PInt`,`Stress`),
250   KEY `TKey_8` (`TKey`,`Align`,`CRight`,`Stress`),
251   KEY `TKey_12` (`TKey`,`Align`,`CCLeft`,`Stress`),
252   KEY `TKey_13` (`TKey`,`Align`,`PInt`,`Stress`),
253   KEY `TKey_9` (`TKey`,`Align`,`CCRight`,`PInt`,`Stress`)
254 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
255
256 --
257 -- Table structure for table `lioba_p_syllable_map`
258 --
259
260 DROP TABLE IF EXISTS `lioba_p_syllable_map`;
261 CREATE TABLE `lioba_p_syllable_map` (
262   `SyllableNum` int(10) unsigned NOT NULL default '0',
263   `PhoneNum` int(10) unsigned NOT NULL default '0',
264   KEY `SyllableNum` (`SyllableNum`)
265 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
266
267 --
268 -- Table structure for table `lioba_p_word_data`
269 --
270
271 DROP TABLE IF EXISTS `lioba_p_word_data`;
272 CREATE TABLE `lioba_p_word_data` (
273   `Num` int(10) unsigned NOT NULL default '0',
274   `TKey` varchar(50) character set latin1 collate latin1_bin NOT NULL
        default '',
275   `TReal` varchar(100) character set latin1 collate latin1_bin NOT NULL
        default '',
276   `CLeft` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
277   `CRight` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
278   `CCLeft` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
279   `CCRight` varchar(5) character set latin1 collate latin1_bin NOT NULL
        default '',
280   `PInt` int(10) unsigned NOT NULL default '0',
281   `PMode` enum('.','?') NOT NULL default '.',
282   `First` int(10) unsigned NOT NULL default '0',
283   `Last` int(10) unsigned NOT NULL default '0',
284   `Align` enum('Manual','Auto') NOT NULL default 'Manual',
285   `LM0` float default NULL,
286   `LM1` float default NULL,
```

```
287   `LM2` float default NULL,
288   `LM3` float default NULL,
289   `LM4` float default NULL,
290   `LM5` float default NULL,
291   `LM6` float default NULL,
292   `LM7` float default NULL,
293   `LM8` float default NULL,
294   `LM9` float default NULL,
295   `LM10` float default NULL,
296   `RM0` float default NULL,
297   `RM1` float default NULL,
298   `RM2` float default NULL,
299   `RM3` float default NULL,
300   `RM4` float default NULL,
301   `RM5` float default NULL,
302   `RM6` float default NULL,
303   `RM7` float default NULL,
304   `RM8` float default NULL,
305   `RM9` float default NULL,
306   `RM10` float default NULL,
307   `AVGF0` float default NULL,
308   `LF0` float default NULL,
309   `RF0` float default NULL,
310   `RMS` float default NULL,
311   `CCLeft2` varchar(100) NOT NULL default '',
312   `CCRight2` varchar(100) NOT NULL default '',
313   `T0` float default NULL,
314   `Ap` float default NULL,
315   `T1` float default NULL,
316   `T2` float default NULL,
317   `Aa` float default NULL,
318   PRIMARY KEY (`Num`),
319   KEY `TKey_2` (`TKey`,`Align`,`CCLeft`,`CRight`,`PInt`),
320   KEY `TKey_3` (`TKey`,`Align`,`CLeft`,`CCRight`,`PInt`),
321   KEY `TKey_4` (`TKey`,`Align`,`CCLeft`,`CCRight`,`PInt`),
322   KEY `TKey_5` (`TKey`,`Align`,`PInt`,`CLeft`),
323   KEY `TKey_6` (`TKey`,`Align`,`CRight`,`PInt`),
324   KEY `TKey_7` (`TKey`,`Align`,`CCLeft`,`PInt`),
325   KEY `TKey` (`TKey`,`Align`,`CLeft`,`CRight`,`PInt`),
326   KEY `TKey_8` (`TKey`,`Align`,`CCRight`,`PInt`)
327 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
328
329 --
330 -- Table structure for table `lioba_p_word_map`
331 --
332
333 DROP TABLE IF EXISTS `lioba_p_word_map`;
```

```
334 CREATE TABLE `lioba_p_word_map` (
335   `WordNum` int(10) unsigned NOT NULL default '0',
336   `SyllableNum` int(10) unsigned NOT NULL default '0',
337   PRIMARY KEY (`SyllableNum`),
338   KEY `WordNum` (`WordNum`),
339   KEY `SyllableNum` (`SyllableNum`)
340 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
341
342 --
343 -- Table structure for table `morpheme_lexicon`
344 --
345
346 DROP TABLE IF EXISTS `morpheme_lexicon`;
347 CREATE TABLE `morpheme_lexicon` (
348   `Orth` varchar(20) character set latin1 collate latin1_bin NOT NULL
        default '',
349   `Trans` varchar(20) character set latin1 collate latin1_bin NOT NULL
        default '',
350   `Class` char(1) character set latin1 collate latin1_bin NOT NULL default '
        ',
351   KEY `Orth` (`Orth`)
352 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
353
354 --
355 -- Table structure for table `phone_table`
356 --
357
358 DROP TABLE IF EXISTS `phone_table`;
359 CREATE TABLE `phone_table` (
360   `Phon` char(5) character set latin1 collate latin1_bin NOT NULL default ''
        ,
361   `Class` char(2) character set latin1 collate latin1_bin NOT NULL default '
        ',
362   `Voicing` char(1) character set latin1 collate latin1_bin NOT NULL default
         ''
363 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Listing D.1: SQL-Anweisungen zur Erzeugung der deutschen Annotationsdatenbank (vgl. Abschnitt 4.1.3).

Anhang E. Auskunft

Dieser Teil des Anhangs enthält ergänzende Informationen zum Telefonauskunftsprojekt (s. Kapitel 6).

E.1. Korpusbestandteile

Das Korpus für die Telefonauskunftssynthese ist aus verschiedenen Gruppen von Sätzen zusammengesetzt.

Die erste Gruppe besteht aus vollständigen Sätzen mit eingebetten Personen-, Orts- und Straßennamen. Die Sätze dieses Typs haben keinen Typnamen, sondern nur eine Satznummer im Typfeld des Dateinamens: (z. B. `s001112_00012_1.wav`).

```
00001: Diese Firma ist eine GmbH
00367: Unser Nachbar heißt Bernd Müller
00464: Das ist die Tochter von Herrn Zimmermann ,
        und seiner Frau M-Punkt
01329: Frau Doktor Bartels geht durch die Großstraße 350 ,
        und die Adresse ist in Bruchköbel
```

Eine weitere Gruppe stellen die MIS-Inventarsätze zur Lautübergangsabdeckung (nach Portele, 1996):

```
048-005:  Der Tischherr, der traditionell auf dem Plüschsofa sitzt,
 die Tischdame und die Tischnachbarn loben in einer Tischrede die
 Kochkünste des berühmten Mannes unter der Kochmütze.
```

Den zahlenmäßig größten Anteil am Korpus haben die Einzelwort-Realisierungen (`''e''`) der Namen in einem prosodischen Schema, das der Telefonauskunftsanwendung entspricht:

```
Vorname Name, Strasse Hausnummer, Ort. (Typ1)
Vorname Name, Strasse, Ort.            (Typ2)

e01197:   Boris Alexander   Janz,   Siedlerweg   33e,   Friesenheim.
```

Für die Ausgabe von Zahlen in der Syntheseanwendung wurden mehrere Aufnahmen durchgeführt:

```
Satztyp 1 (z-1-pro-[0-9]):
Eine Zahl (von "Null" bis "Neun"),
die isoliert mit steigender Intonation ausgesprochen wird.

Satztyp 2 (z-1-fin-[0-9]):
Eine Zahl (von "Null" bis "Neun"),
die isoliert mit fallender Intonation ausgesprochen wird.

Satztyp 3 (z-1-profin-[0-9]u[0-9]):
Zweimal die gleiche Zahl (von "Null" bis "Neun"),
die zunächst mit steigender und anschließend
mit fallender Intonation ausgesprochen wird.

Satztyp 4 (z-2-pro-[0-9]u[0-9]):
Zwei Zahlen (von "Null" bis "Neun") als Paar realisiert,
wobei die erste monoton und die zweite mit steigender
Intonation ausgesprochen wird.

Satztyp 5 (z-2-fin-[0-9]u[0-9]):
Zwei Zahlen (von "Null" bis "Neun") als Paar realisiert,
wobei die erste monoton und die zweite mit fallender
Intonation ausgesprochen wird.

Satztyp 6 (z-2-profin-[0-9]u[0-9]):
Zwei identische Zahlenpaare (von "Null" bis "Neun"),
wobei das erste Element des Paares
jeweils monoton ausgesprochen wird,
und das zweite zuerst mit steigender Intonation
und im zweiten Paar mit fallender.

Satztyp 7 (zv00[0-1][0-9][0-9]):
Drei Zahlen (von "Null" bis "Neunundneunzig"),
die ersten beiden mit steigender,
die letzte mit fallender Intonation ausgesprochen.

Satztyp 8 (zp0[0-9][0-9]):
Sechs Zahlen (von "Null" bis "Neun") in der Reihenfolge:
```

```
monotone, steigende, monotone,
steigende, monotone und fallende Intonation (drei Zahlenpaare, zp)

Satztyp 9 (ze0[0-1][0-9]):
Drei einzelne Zahlen (von "Null" bis "Neun"),
die ersten beiden mit steigender,
die letzte mit fallender Intonation gesprochen.
```

Alphabeteinheiten:

```
Buchstabiereinheiten, in "alphabet-medial" einzeln und medial;
 in "alphabet-profin" paarweise (steigend, fallend).

alphabet-medial:
 A B C D E F G Punkt. H I J K L M N O P Punkt.
 Q R S T U V  Punkt. W X Y Z Ä Ü Ö ß Punkt.

alphabet-pro-fin:
M, Y.

alphabet-pro-fin-rest:
Ä, I.

Alternative Aufnahmen:

alphabet2-medial:
A M.

alphabet2-profin:
A, M.
```

Verschiedene Ansagetexte für die Telefonauskunft:

```
ansagen1:
Die Rufnummer lautet: Punkt
```

Sonderrufnummern: Rufnummern mit festgelegter Aussprache.

```
sonderrufnr1:
null achthundert,              null hundertachtzig,
null hundertneunzig,           null neunhundert,
Punkt.
```

Firmennamen: Die Namen bzw. Namensbestandteile wichtiger Firmen und Institutionen, sowie Namenszusätze.

HardSoftWare: verschieden stark eingedeutschte Aussprachevarianten von Hard- und Software, Computer und Service.

```
firmen-zusatz-01: Deutsches Rotes Kreuz, Krankenhäuser,
 Krankenhaus, Klinik, Kliniken, Hospital, Punkt.
```

Einheiten zur Ergänzung des engl. ⟨th⟩ in verschiedenen Kontexten (Die Aussprache der Kontexte ist bewusst z. T. eingedeutscht).

```
th-kontexte:
than, those, the, their, they, this, these, thus, Punkt.
thank, thaw, theatre, theft, thermal, thick, Punkt.
thigh, thong, thorn, thunder, thousand, through, Punkt.
throw, thread, three, thrift, thrive, thrust, thwart, Punkt.
```

Ergänzende Multiphoneinheiten, eingebettet in einen Trägersatz. Die relevanten Einheiten befinden sich jeweils im dritten Wort. Die Trägersätze sind an Portele (1996) angelehnt.

```
ergaenzungen-(Nr):
m"an  "vE:.n@s ?9:.t@."laI  g@.t"a:n.
```

E.2. Abdeckungsstatistiken zum Korpus

Kategorie	Vornamen	Nachnamen	Straßen (progredient)	Straßen (medial)	Orte
Abdeckung Instanzen vorher	81,6%	36,5%	35,6%	35,4%	74%
Abdeckung absoluter Instanzen vorher	27658960	13513925	11858596	11784144	28354994
Abdeckung Instanzen nachher	80,9%	30,7%	35,4%	35,1%	73,4%
Abdeckung absoluter Instanzen nachher	27407289	11383272	11792067	11676847	28118042
Abdeckung Typen vorher	17,15%	1,17%	3,67%	3,29%	11,05%
Abdeckung absoluter Typen vorher	78813	31728	14033	12560	1901
Abdeckung Typen nachher	16,92%	1,02%	3,52%	3,19%	10,95%
Abdeckung absoluter Typen nachher	77719	27735	13445	12171	1885
Typen	459428	2715333	382118	382118	17207
Instanzen	33880414	37026568	33283203	33283203	38299158

Tabelle E.2.1.: Abdeckung der Volltexteinträge in der Projektdatenbank durch die Worteinheiten im klickTel-Korpus, sortiert nach Eintragsarten und verschiedenen Phrasenpositionen (vor und nach dem Pruning) — vgl. Abschnitt 6.1.5.

E.3. Abdeckungsstatistiken zum Eigennamenlexikon

	Vornamen	Namen	Straßen	Orte	Gesamt
Anzahl Instanzen (Volleintrag)	33.880.399	37.026.568	33.283.203	38.299.158	142.489.328
Anzahl transkribierter Instanzen (Volleintrag)	33.114.859	24.751.678	25.088.255	37.796.946	120.751.738
Anteil transkr. Instanzen (Volleintrag)	97,7%	66,8%	75,4%	98,7%	84,7%
Unterschiedliche Volltexte	459.427	2.715.332	382.118	15.311	3.484.258
Unterschiedliche transkr. Volltexte	167.697	111.981	66.027	9.656	347.704
Unterschiedliche Transkriptionen (Volltexte)	136.169	105.917	64.430	9.382	308.187
Unterschiedliche Teileintragstexte	190.006	1.244.876	284.739	15.270	1.580.971
Unterschiedliche transkr. Teileintragstexte	8.751	64.182	41.744	9.653	105.046
Unterschiedliche Transkriptionen (Teileinträge)	7.006	50.679	40.076	9.532	89.228

Tabelle E.3.1.: Abdeckungsstatistiken für die Voll- und Teileinträge der klickTel-Projektdatenbank durch die Transkriptionen im Aussprachewörterbuch (vgl. Abschnitt 6.2.5).