Helsinki University of Technology Laboratory of Acoustics and Audio Signal Processing
Espoo 2001 Report 63

OBJECT-BASED MODELLING FOR REPRESENTING AND PROCESSING SPEECH CORPORA

Toomas Altosaar

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory of Acoustics and Audio Signal Processing
Espoo 2001 Report 63

OBJECT-BASED MODELLING FOR REPRESENTING AND PROCESSING SPEECH CORPORA

Toomas Altosaar

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission for public examination and debate in Auditorium S4, Department of Electrical and Communications Engineering, Helsinki University of Technology, Espoo, Finland, on the 28th of September, 2001, at 12 o'clock noon.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Laboratory of Acoustics and Audio Signal Processing

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Akustiikan ja äänenkäsittelytekniikan laboratorio

Helsinki University of Technology
Laboratory of Acoustics and Audio Signal Processing
P.O.Box 3000
FIN-02015 HUT
Tel. +358 9 4511
Fax +358 9 460 224
E-mail lea.soderman@hut.fi

ISBN 951-22-5622-3
ISSN 1456-6303

Otamedia Oy
Espoo, Finland 2001

Object-Based Modelling for Representing and Processing Speech Corpora

Toomas Altosaar

Abstract

This thesis deals with modelling data existing in large speech corpora using an object-oriented paradigm which captures important linguistic structures. Information from corpora is transformed into objects and are assigned properties regarding their behaviour. These objects, called speech units, are placed onto a multi-dimensional framework and have their relationships to other units explicitly defined through the use of links. Frameworks that model temporal utterances or atemporal information like speaker characteristics and recording conditions can be searched efficiently for contextual matches. Speech units that match desired contexts are the result of successful linguistically motivated queries and can be used in further speech processing tasks in the same computational environment. This allows for empirical studies of speech and its relation to linguistic structures to be carried out, and for the training and testing of applications like speech recognition and synthesis.

Information residing in typical speech corpora is discussed first, followed by an overview of object-orientation which sets the tone for this thesis. Then the representation framework is introduced which is generated by a compiler and linker that rely on a set of domain-specific resources that transform corpus data into speech units. Operations on this framework are then presented along with a comparison between a relational and object-oriented model of identical speech data.

The models described in this work are directly applicable to existing large speech corpora, and the methods developed here are tested against relational database methods. The object-oriented methods outperform the relational methods for typical linguistically relevant queries by about three orders of magnitude as measured by database search times. This improvement in simplicity of representation and search speed is crucial for the utilisation of large multi-lingual corpora in basic research on the detailed properties of speech, especially in relation to contextual variation.

Keywords: speech corpora, speech database, object-oriented model, database access, speech processing.

Preface

The work leading up to this thesis was carried out during the years 1985-2001 in the Laboratory of Acoustics and Audio Signal Processing, part of the Helsinki University of Technology, situated in Espoo, Finland. I am grateful to my supervisor, Prof. Matti Karjalainen, for organising the long-term financial support required for this research. I thank him for many things, especially for teaching—and instilling in me—the desire to model real-world objects using an object-oriented approach. It has been a privilege to work with this man of great intellect who possesses an unquenchable thirst for knowledge in so many different fields.

The financial support of the Academy of Finland, SITRA (the Finnish National Fund for Research and Development), and TEKES (the Finnish National Technology Agency) is gratefully acknowledged.

I would like to thank Martti Vainio from the Department of General Linguistics, University of Helsinki, for his collaboration. Without him the system would not have evolved in its current direction and many challenging and interesting phonetic phenomena would have remained unexplored. Also, I express my thanks to Bruce Millar from the Australian National University, for his cooperation. Jim Hieronymus from the NASA Ames Research Centre and Christoph Draxler from the Department of Phonetics and Speech Communication, University of Munich, have given me valuable advice and new insight regarding this work. I thank both of them for many interesting discussions and useful suggestions. I have had the pleasure of working with Einar Meister, from the Technical University of Tallinn, on several problems dealing with my native Estonian language. Both Nina Alarotu and Mietta Lennes, from the Department of Phonetics, University of Helsinki, have played their part in making this work more palatable by manually annotating large portions of speech.

From an early age I have been interested in human communication. Technically, amateur radio offered me a path to explore both wireless telegraphy and telephony in the mid 1970's with the guidance of Keith Baker, from Beaconsfield, Québec. Ten years later Douglas O'Shaughnessy, from McGill University, was first to formally teach me what speech was and how it could be processed. Ever since I came to Finland in 1985 I have found the dynamic atmosphere of the Acoustics Laboratory to be an ideal place to conduct research. Unto Laine and Paavo Alku have given me good advice over the years. Lea Söderman, our laboratory's secretary, has helped to keep the level of bureaucracy to a minimum. I also thank the many other members of the lab, past and present, for their always willing assistance.

I am also thankful for the support that I have received from family and friends. Words cannot express the gratitude that I feel for my parents Laine and Heino. They have always supplied love and encouragement and looked favourably upon my decision to pursue further studies—albeit away from my home in Canada. My three older brothers have shaped my life favourably: Peeter has taught me humanity, Erik a profession, and Illimar has broadened my viewpoint. To Endel Ruberg—who through the Estonian epic of Kalevipoeg first ignited the spirit of Finland in me, Mike Riggin—for teaching me that the work in this world is accomplished by men who wear white shirts and black shoes, Martin Pflug—for many adventures in the great outdoors, and to the many other Canadians, I extend my thanks for influencing my life in a positive manner.

Many friends here in Finland, such as Sinikka and Antti Vuorinen, have helped make Finland a second home for me, as have Philip Vara, Kalervo Rinne, Antti Marjamäki, and Antti Arjas. I'd especially like to thank Georg Metsalo for over 20 years of companionship, for reminding me on a nearly daily basis that "it always takes two more years" to complete a thesis, and also his family for their care and understanding. Jean-Luc Olivès and Essi Sirviö have proven to be good friends both in and out of the swimming pool. Many other people, too numerous to mention, have helped me through the good as well as difficult times. I am grateful to them all.

Finally, I thank my wife Katrina for her love, wisdom, patience, and enduring smile. Our children Johanna and Oskar have brought much joy into our lives. For all these gifts of life I am thankful.

Kukunta, Espoo
September 16, 2001

Toomas Altosaar

Table of Contents

List of Abbreviations

AIFF	Audio Interchange File Format
ANDOSL	Australian National Database of Spoken Language
ANN	Artificial Neural Network
BAS	Bayerisches Archiv für Sprachsignale
CBN	Critical-Band Number
CLOS	Common Lisp Object System
CSM	Corpus Specification Model
ELRA	European Language Resources Association
Finnbase	Finnish Speech Corpus and Database based on QSSDB
GRF	Generic Representation Format
Kiel	Kiel Corpus of Read Speech
LDC	Linguistic Data Consortium
NIST	National Institute of Standards and Technology
NSLD	Notional Spoken Language Description
OO	Object-Oriented
OOP	Object-Oriented Programming
OO-DBMS	Object-Oriented Database Management System
QSSDB	QuickSig Speech Database System
QuickSig	QuickSig Digital Signal Processing System
RDBMS	Relational Database Management System
SAMPA	Speech Assessment Methods Phonetic Alphabet
TENN	Time-Event Neural Network
TIMIT	DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus
WAV	WAVE PCM File Format
XML	Extensible Markup Language

1 Introduction

Human communication is largely based on speech. With recent advances in computing machines and spoken natural language theories, many attempts have been made at improving the man-machine interface. Since speech is a complex phenomenon that escapes a clear algorithmic description, large collections of annotated speech have been created to better understand aural human communication.

Annotations of speech provide a crucial starting point from which further studies and analyses on the phenomena of spoken language can be initiated. Speech corpora and databases may provide sequential, atemporal, and time-aligned annotations for some or all of their signal recordings. These descriptions of the speech process, symbolic in nature and typically consisting of textual information, offer a detailed account of the events as well as the environment in which they occurred. For example, a practical requirement for comprehensive phonetic studies and robust speech technology applications is fulfilled by time-aligned annotations, which allow for determining segment locations within a speech waveform.

However, potential use of available data is limited unless a model exists that matches the detail and depth of the annotations. In general, a speech corpus does not advocate or supply a model for its annotations and it remains up to the user to design and implement one before the data can be analysed. On the other hand, speech databases may contain a model that is suitable for its native set of corpora expressed in a strict and homogeneous format. Therefore, if foreign corpora are to be studied in a database system, they must first be transformed into an acceptable format. Unless an accurate method is available for this transformation, the data may lose much of its inherent detail.

Database queries are frequently accomplished by reading in annotations consisting of label strings and then searching for certain character combinations through the use of a linear string processing language. By relying on a purely symbolic and textual representation, a superficial model of speech is formed which often fails to capture the intricacies occurring in spoken language. Thus, the user faces an overly simple interface to the annotation data forcing queries to become complex, difficult to formulate, and therefore prone to errors. As speech databases become larger, the efficiency of searches becomes an issue of great importance.

Results of database queries are typically textual in nature as well. For example, a phone within a signal may be represented in printed form as a signal file directory and name followed by two integer indices (indicating the offsets within the signal file) and then followed by its phonetic symbol expressed as a series of characters. Such a representation is a very depleted expression of the actual phone that was uttered since it does not contain any additional knowledge that might be useful in later analysis

programmes operating within or outside of the database system. When analyses are performed outside of the database the results need to be first exported, e.g., via the file system. This added overhead may preclude some time-critical applications, e.g., speech synthesis where context searches over a corpus are performed in real-time.

1.1 Aim of the Thesis

This thesis presents a methodology for modelling speech units and their relationships explicitly rather than leaving the representation entirely on a symbolic and textual level. Based on object-oriented programming (OOP), speech units are modelled as objects that contain inherent knowledge allowing them to behave intelligently. For example, modelling all possible human speech sounds using a set of class-based phonetic features allows phones and phonemes to be aware of their constituent distinctive features enabling database searches to freely exploit their class inheritances.

Other speech units—such as sentences, words, and syllables—are also modelled as objects. These units are aware of their immediate contextual environment through explicit links, e.g., a phoneme *x* that is linked to its neighbouring phonemes is able to determine its sequential order within the phoneme level of the linguistic domain it resides in. Likewise, a link in the hierarchical dimension indicates to *x* which syllable it belongs to. Units of different types can be linked freely with one another too, e.g., a link from a phoneme to a phone can be used to indicate its spoken realisation. Relationships between units that are not located in the immediate vicinity of one another and behave in a regular manner—according to some defined model—can be expressed through computational links. Finally, by including objects derived from atemporal annotations, a contribution towards a more complete representation of reality is made.

This thesis develops a two-level abstraction technique to transform symbolic speech annotations for an utterance into a multi-dimensional linked model called a *representation framework*. The first level forms a generic representation where data from speech corpora are reinstated with missing type information. The second level uses the generic representation along with a set of resources to build a representation framework that provides a flexible and useful foundation on which to conduct analyses. Database queries can thus be formulated in a compact and robust manner since a speech unit's dependencies, internal characteristics, and its location in the linked framework are defined within the model rather than in the search expression.

The frameworks are generic in the sense that they are independent of many factors such as phonetic alphabet, language, and character set. Furthermore, different transcription styles, e.g., linear, non-

linear, with or without overlap, and styles that support componential features can also be represented in the same framework. Regardless of the corpus under study, database access becomes uniform thereby facilitating comparative studies of spoken language. Therefore, frameworks provide the potential of establishing quantifiable equivalence relationships between different underlying annotation formalisms found in existing corpora, as well as making possible the comparison of competing language-theories—all within the same computational environment.

The richness of the symbolic description of speech, i.e., the amount of detail that is supplied with an utterance, determines largely what structures can be created. For example, acoustic, phonetic, linguistic, orthographic, and prosodic structures can be formed if these information sources exist within a corpus. If required, some additional structures can be added by computational means, e.g., syllabification of words using a lexicon or rule-based system, pitch calculated from the acoustic waveform, etc.

Database searches over these linked structures are performed by pattern matching. Functions are used to define search templates that traverse the frameworks and their corresponding substructure seeking out desired contexts. Objects within the structures can be tested against their local and class inherited properties as well as their relations to other objects. A query returns as a list the matching objects within a framework. Since actual objects are returned—not simply textual representations—a search returning a set of phone objects can have their pitch contours calculated immediately since each phone is linked to the speech waveform and is aware of its temporal extent. These contours can then be used in the same computational environment, e.g., in training neural networks that generate microprosody in a speech synthesiser.

Corpus based speech research requires efficient searches over linguistically motivated structures. A vowel in a context study might require 18,000 searches of a database to produce one parameter scatter plot. Since many of the searches might require even larger contexts, several levels of abstraction may have to be searched. The object-oriented database structure developed in this thesis allows these searches to be fast and efficient with respect to memory usage.

1.2 Organisation of the Thesis

The thesis is organised as follows. First, an overview of speech corpora and databases is presented that describes their typical contents, the varying standards employed, and the plethora of formats used to represent data. A critical overview then follows of some current speech database environments where existing weaknesses related with these approaches are determined. This thesis integrates theory,

concepts and models from:

i. the computer sciences in the form of object-oriented programming (OOP),
ii. linguistics and phonetics,
iii. signal processing,
iv. and database technology.

Therefore, section 3 introduces the concepts related to each field and the QuickSig Speech Database (QSSDB) system on which the described methodology was developed. The latter is presented in more detail and lays down a foundation for the integrated object-oriented database and speech processing environment. In section 4 the concept of a *representation framework* is developed where information from speech corpora can be placed. Since databases exist in a diverse variety of formats, a formalism is described that models the storage of speech material existing in files and is presented in section 4.2. Then the resources and methods used to interpret and compile heterogeneous information from speech corpora into generic representation units is covered. Building hierarchical structures from a variety of different types of representation units using a linker is then developed and visualisation of the frameworks then follows. Database queries, performed by traversal of these structures by pattern matching functions, are presented in section 5. The penultimate section covers applications where QSSDB has been used, the strengths and weaknesses of the system, and possible future development. Finally, the main points of this thesis are recapitulated in the summary.

1.3 Author's Contribution

The research process and development work that has gone into this thesis has been a collaborative effort over the past 15 years. Matti Karjalainen pioneered and developed the QuickSig environment from 1987 onwards along with the help of the author and Paavo Alku, and planted many of the initial ideas and concepts which have evolved and expanded into the current QSSDB system. From 1996 the expansion has largely taken place with phonetic knowledge and useful ideas supplied by Martti Vainio.

The author has been the main designer and implementer of the QSSDB system. QSSDB extends the modelling accuracy of QuickSig when dealing with annotations regarding spoken language. The design philosophy of QSSDB is unique when compared to other systems that process speech corpora and annotations, and in many cases is superior. For example, a thorough object-oriented modelling approach is adhered to which allows for processing different corpora in a generic and scalable framework. Efficient database access, when compared to a commonly used relational database model of speech, is achieved. Visualisation of temporal and atemporal spaces using a 3-dimensional browser for

utterances and corpora helps promote a better understanding of the information at hand, their relationships, and the generated models of speech. The author is responsible for these major improvements and insights.

In this thesis part of the ANDOSL corpus is modelled with a RDBMS and an OO–DBMS for comparative purposes. Sub-sections that deal with RDBMS modelling, i.e., 3.4.1 and 5.3.1-5.3.3, were written in collaboration with Bruce Millar since he carried out the modelling and tests in the RDBMS. These tests demonstrated the superiority of QSSDB relative to a relational database query system both in terms of complexity and search time.

Some of the ideas developed in this thesis have been originally published in publications where the author has been involved (Altosaar & Karjalainen, 1987; Karjalainen, Altosaar & Alku, 1988; Altosaar & Karjalainen, 1988; Karjalainen & Altosaar, 1988; Karjalainen, Altosaar, Alku, Lehtinen & Helle, 1989; Altosaar & Karjalainen, 1989; Laine & Altosaar, 1990; Altosaar & Karjalainen, 1991a; Karjalainen & Altosaar, 1991; Altosaar & Karjalainen, 1991b; Laine, Karjalainen & Altosaar, 1991; Altosaar & Karjalainen, 1991c; Karjalainen, Välimäki, Altosaar & Helle, 1992; Karjalainen & Altosaar, 1992; Altosaar & Karjalainen, 1992; Karjalainen & Altosaar, 1993; Laine, Karjalainen & Altosaar, 1994; Altosaar, Karjalainen & Vainio, 1994; Vainio, Altosaar, Karjalainen & Iivonen, 1995; Altosaar & Meister, 1995; Altosaar, Karjalainen & Vainio, 1996a; Vainio & Altosaar, 1996a; Altosaar, Karjalainen & Vainio, 1996b; Vainio & Altosaar, 1996b; Altosaar, Vainio & Karjalainen, 1996; Vainio, Aulanko, Altosaar & Karjalainen, 1997; Karjalainen, Boda, Somervuo & Altosaar, 1997; Alarotu, Lennes, Altosaar, Malm & Karjalainen, 1997; Karjalainen, Altosaar & Vainio, 1998; Altosaar, Karjalainen, Vainio & Meister, 1998; Karjalainen & Altosaar, 1998a; Vainio, Altosaar, Karjalainen & Aulanko, 1998; Karjalainen & Altosaar, 1998b; Karjalainen, Altosaar & Vainio, 1998; Altosaar & Vainio, 1998; Vainio & Altosaar, 1998; Karjalainen, Altosaar & Huttunen, 1998; Olivés *et al.*, 1999; Altosaar, Vainio & Karjalainen, 1999; Vainio, Altosaar, Karjalainen, Aulanko & Werner, 1999; Altosaar, Millar & Vainio, 1999; Meister, Eek, Altosaar & Vainio, 1999; Meister, Eek, Altosaar & Vainio, 2000; Altosaar & Vainio, 2000; Vainio & Altosaar, 2000; Altosaar, Karjalainen & Vainio, 2001).

2 Speech Corpora and Databases

In this thesis the terms *speech corpus* and *speech database* are differentiated; the former indicates a collection of speech tokens along with some additional information such as annotations. The latter refers to speech corpora along with a mechanism to access the tokens, e.g., a software system (Hendriks, 1990). Speech tokens usually consist of acoustical waveform recordings of one or more speakers in certain environments. Additionally, these collections of spoken language may contain other related information such as orthographic transliterations, phonemic transcriptions and segment temporal intervals marked with their broad and narrow phonetic labels. Stress markers, surface structures, and deep structures may also be available for representing prosodic, syntactic, and semantic information, respectively. These annotations are high-level symbolic representations of the waveform and enable speech data to be retrieved according to desired search criteria. In a database environment, annotations can be used to find segmental and prosodic regularities of languages, to study dialectal varieties, and to investigate speaking styles (Kohler, 1998). Other signals captured during the recording session, such as electroglottograph waveforms for accurate pitch determination and video for capturing facial movements, may exist also.

2.1 Importance of Speech Corpora and Databases

Speech corpora and databases have become a cornerstone of spoken language technology. They provide a critical and necessary resource that applications such as speech analysis, synthesis, and recognition systems need to utilise in order to become robust. Using this resource, speech scientists can search for and identify the invariance within speech—collected from a large number of speakers from a variety of recording environments, differing dialectal regions, and linguistically diverse areas—to gain a better understanding of human aural communication. Developers of speech technology products can refine their market offerings by first testing them thoroughly on large volumes of data.

Within the past two decades the speech community has realised the importance of having access to large collections of carefully recorded, transcribed, segmented, and labelled speech data for the development of spoken language technology. Some corpora have been utilised extensively by independent research groups thus becoming de facto standards for developing, testing, and comparing the performance of speech processing tasks. One such speech corpus that was first released in 1988 and still enjoys widespread use is TIMIT (Garofolo *et al.*, 1993). It contains speech from 630 American English speakers each uttering 10 sentences that were designed to express dialect, phonemic-compactness, and phonetic diversity. The EUROM1 series of speech corpora (Chan *et al*., 1995) represents a collective effort to cover many of the European languages. Spoken language corpora for several Asian languages also exist, e.g., JEIDA, a 40 volume CD-ROM set for Japanese (Itahashi, 1986). The Australian

National Database of Spoken Language (ANDOSL) is a 30 CD-ROM collection of speech covering basic lexical entities, phonetically-rich sentences, and a semantically-constrained spontaneous speech task, in which 264 speakers participated (Millar, Vonwiller, Harrington & Dermody, 1994). Minority languages have had databases created for them as well, e.g., the Estonian Phonetic Database (Meister & Eek, 1999), which is being used among other things to study the three degrees of quantity the language offers. Finnbase, containing over 5000 word and sentence tokens from a small set of Finnish speakers has been used in several application areas (Altosaar, Karjalainen, Vainio & Meister, 1998).

Large collections of digitised data represent an indispensable resource for spoken language studies. Corpora have been produced for several of the world's languages and more are currently being designed and produced for a variety of applications. Central repositories for speech corpora exist, e.g., the LDC (Linguistic Data Consortium, 2000) and ELRA (European Language Resources Association, 2000).

2.2 Contents of Speech Corpora

Speech corpora often exist as collections of recorded signal files accompanied by one or more parallel files containing symbolic and/or numeric data. For example, associated with each audio file in the TIMIT corpus are three files: an orthographic transcription, a time-aligned word boundary transcription, and a time-aligned phonetic transcription labelled with symbols from the ARPABET phonetic alphabet. In the Kiel Corpus of Read and Spontaneous Speech published in 1995 (Simpson, Kohler & Rettstadt, 1997), the sentence, word, and phone levels are supplied as in TIMIT but exist within a single file using a different format. The phonetic alphabet used is German SAMPA, a modified and augmented SAMPA (Speech Assessment Methods Phonetic Alphabet) (ESPRIT, 1987) notation tailored for the German language. Other differences also exist: first, the canonical form of utterances is available indicating standard pronunciation. Secondly, the phone level contains symbols to indicate uncertain beginnings, quantity, laryngealisation, mispronunciation, insertion, deletion, and replacement. Other contextual properties are indicated as well: the beginning and end of sentences, phrases, and words, the locations of primary and secondary stress, and whether a word is a function word or part of a compound word. Table 1 indicates the symbolic information supplied with the spoken utterances of five different exemplary speech corpora. Additional useful information may also be supplied with these corpora such as in ANDOSL (Millar *et al.*, 1994) where extensive information regarding speakers and recording environments is available.

Table 1. Some primary information available in five different speech corpora.

Information Type				Corpus				
				ANDOSL YB Sentences	Estonian Phonetic DB	Finnbase	Kiel Read Speech Vol. 1	TIMIT
Temporal	Numeric	Acoustic	Air Pressure	NIST, 20 kHz	raw, 20 kHz	AIFF, 22.05 kHz	NIST, 16 kHz	raw, 20 kHz
			Pitch	–	–	AIFF, 200 Hz	–	–
			Loudness	–	–	AIFF, 200 Hz	–	–
	Symbolic	Orthographic	Sentence	+	+	time-aligned	+	time-aligned
			Word	–	–	time-aligned	–	time-aligned
		Linguistic	Word	–	–	–	German SAMPA canonical	ARPABET canonical, lexicon
			Phoneme	MRPA time-aligned	Estonian SAMPA time-aligned	+	German SAMPA canonical	ARPABET canonical, lexicon
		Phonetic	Sentence	–	–	–	+	–
			Phrase	–	–	–	+	–
			Word	–	–	time-aligned	+	–
			Compound Word	–	–	–	+	–
			Phone	–	–	Worldbet, Finnbet time-aligned	German SAMPA time-aligned	ARPABET time-aligned
			Segment	–	–	Worldbet, Finnbet time-aligned	–	ARPABET time-aligned
		Prosodic	Primary Stress	–	+	–	+	+
			Secondary Stress	–	+	–	+	+
Atemporal				extensive	–	fair	–	minimal

legend: + supplied by corpus; – not supplied by corpus

Different annotation styles have been employed to annotate speech. The type of phonetic segmentation used in the TIMIT corpus is linear without overlap, meaning that no intersection of phones occur temporally. However, at the word level, phones may be shared at word boundaries, e.g., when gemination occurs (Garofolo *et al.*, 1993). The Kiel corpus also employs a linear segmentation without overlap approach to phonetic segmentation and labelling. In addition, it supports componential residues—the ability to indicate that some remnants of a deleted phone still exist, e.g., nasalisation. This notion of complementary phonology combines the advantages of linear base forms for lexical database searches with the need for contrastive phonetic adequacy (Kohler, 1998). Other types of annotation styles can also be formulated and used for symbolically describing speech, e.g., specifying the domain of articulatory or acoustic properties across traditional segmental boundaries (Kelly & Local, 1989). The ability to transform one annotation style into another, when possible, would be beneficial since it would allow speech to be viewed from different symbolic perspectives.

2.3 Difficulties Associated with Accessing and Utilising Speech Corpora

When fully utilised, a wealth of knowledge regarding spoken language can be inferred from speech corpora. However, extracting information from the implicit relationships that exist within a corpus can be difficult due to the unrelated way in which data is usually stored and represented. Relationships between waveform and symbolic data often remain implicit in the file structures found on the corpus' distribution media and data must be parsed and interpreted before being put to any significant use. For example, the large number of annotation details associated with speech signals in the Kiel corpus would make forming a new representation framework—every time a search was initiated—computationally expensive. Representing the structure algorithmically alone, i.e., without any structure, would not only be expensive but prone to errors since complex relationships would be difficult to express. Repeated computational expenses could be avoided by implementing a model in which individual objects can be automatically stored and retrieved. This type of behaviour can be efficiently accomplished in the object-oriented (OO) paradigm using persistent databases. Furthermore, due to the existence of different audio file formats, computer readable phonetic alphabets, and database standards, the use of different speech corpora in a common analysis environment has not always been feasible thus hindering cross-lingual studies. Finally, relational databases are not necessarily the most intuitive and flexible manner in which to describe and analyse speech (Altosaar, Millar & Vainio, 1999). An explicit, object-oriented, richly detailed, and persistent model of speech would offer more opportunities for exploring the complex processes existing within spoken language.

2.4 Overview of Earlier Speech Database Development

Accessing and further processing of speech corpora are typically carried out by using commercially available speech processing environments or by research groups developing their own software tools. A system that was commercially available and still enjoys widespread use is `esps waves+` (Entropic, 1999) along with additional packages such as `htk` that allows users to pursue speech recognition strategies using Hidden Markov Models. The labelling and segmentation tool `aligner` is also available in `esps waves+` for a subset of languages such as English.

Several research groups have developed their own systems for dealing with speech corpora. DEMSI (Hendriks, 1990) is one such system that was designed to allow for detailed acoustic-phonetic research. The formalism permits accessing speech databases at a high level of abstraction allowing users to phrase queries in domain-specific terms. XASSP (Simpson *et al.*, 1997), actively developed at the Institute of Phonetics at Kiel University since 1979, is a programme package that is used for segmentation, labelling, and analysing speech data independently of language.

EMU (Cassidy & Harrington, 1996), developed at Macquarie University, is a speech data management system designed for labelling, managing, and retrieving data from speech corpora via queries. It has been applied to databases of Australian English with extensions made to process Chinese as a goal in making EMU language independent (Cassidy, 1999). Queries in EMU can include constraints involving both sequential and hierarchical relations within an utterance. However, some segmental constraints in EMU are difficult or impossible to express and additional scripts may need to be written to extract desired segments (Cassidy & Harrington, 1996). The results of queries performed in EMU are processed separately by exporting them to one of several available analysis tools via text files. Depending on the speech processing task at hand, this may be a limiting factor in some real-time applications since direct and immediate access to the speech waveform is not straightforward due to the intermediate textual representation of the results. However, by using active database references it may be possible to extract data immediately and use it within the same environment (Cassidy, 1998). EMU supports multiple hierarchies, where a hierarchy is made up of levels such as words, syllables, and phonemes. A specified level in any one single hierarchy can participate as a level in another intersecting hierarchy. The strength of this approach is memory efficiency since redundancy is supposedly eliminated—at the cost of a true and detailed model. However, this approach may exhibit weakness in its modelling architecture since logically different data types are grouped together into the same object, thus a) forcing a loss of model accuracy, and b) inducing database access to become reliant on a relational database management system (RDBMS) formalism where indexing is a prerequisite. For example, it remains unclear how the canonical form of speech expressed as phonemes and the realised

speech as phones could be effectively represented using equivalent objects.

The Festival system (Black, Taylor, Caley & Clark, 2000), developed at the University of Edinburgh, was primarily designed for speech synthesis studies but has been extended to handle speech corpora. Developed by the authors of Festival are Heterogeneous Relation Graphs (HRG) as a method for representing linguistic information (Taylor, Black, & Caley, 2001). HRGs use typed co-indexed relations to specify structure as in EMU.

A model called Annotation Graphs (AG) (Bird & Liberman, 2001) has also been suggested as a flexible and extensible architecture for modelling linguistic annotations. Similar to the HRG model, AGs are directed acyclic graphs where each arc in the graph has a symbolic property associated with it (the label) while some of the nodes may have time stamps relating the information to locations within signal data. The ATLAS system (Bird *et al*., 2000) combines the physical layer (where data is stored) and the application layer (where the data is put to use) with an intermediate logical layer to support the annotation graph formalism. QSSDB allows for the same physical, logical, and application levels to exist; furthermore in QSSDB the logical level can be formed when required automatically, e.g., by rule-based systems, thus permitting an extra degree of freedom. A weakness of the ATLAS system is that the temporal markers are not linked, so that changing a phone boundary requires a separate step to change the word, phrase, and sentence boundary.

When comparing the work presented in this thesis to the above mentioned systems, the major differentiating factor is the ability of QSSDB to efficiently model speech from different corpora in the same computational environment. By making use of the object-oriented paradigm—that is used throughout QSSDB—a more natural and flexible model of speech is formed permitting exploration and more intuitive studies to be carried out.

3 General Methodology

This chapter introduces some prerequisite information used to develop the concept of a representation framework presented later on in this thesis. To set up the perspective, object-oriented methodology is first introduced followed by signal representation and processing techniques. A short introduction to linguistics and its branch of phonetics is then given. Persistent storage issues are then covered with a review of relational and object-oriented database management systems, keeping in mind performance issues that can be measured via impedance mismatch. Finally, the QuickSig signal processing environment is presented on which the research presented in this work was conducted on.

3.1 Object-Orientation

Object-oriented programming (OOP) is a paradigm that is being used extensively in current computer technology. Defining a set of communicating entities called *objects*, OOP provides a foundation onto which systems modelling real-world phenomena can be built. An *object model* describes objects and their relationships while the operations associated with objects and their effects is described by a *functional model*. Models are abstractions of reality, which attempt to capture essential information about a system. However, modelling is not merely recording reality; it is a creative process—the abstractions have to be defined and the structure of the model has to be created (Awad, Kuusela & Ziegler, 1996). Speech and language are both complex phenomena and are well suited to being modelled with an object-oriented approach.

Real-world knowledge can be flexibly modelled and embedded into OOP software due to the structural framework that it provides. Complex systems can be designed by breaking them down into their components. Each subsystem is modelled separately by packaging and hiding its internal behaviour from an outside perspective and has an interface to the external world that permits its independent testing and verification. Correct distribution of knowledge throughout a system enables large integrated software systems to become feasible.

The inclusion of an object formalism in a computer language can be accomplished by enriching it with an infrastructure that permits operating on structured data in a natural and intuitive manner. Computer languages supporting an OOP formalism can either have the paradigm included during their design stage, or have an OOP extension added to them later on. Languages such as C++ (Stroustrup, 1997), Smalltalk (Aoki, 2000), Java (Preiss, 1999), and Common Lisp (Steele, 1990; Keene, 1989) with its Common Lisp Object System extension (Lisp/CLOS), support OOP. The terminology used to define both *object* and *function* models in these different languages vary. Since Lisp/CLOS is the

implementation language of the system described in this work, a viewpoint from the Lisp programmer's perspective is taken and simple code examples are given below to characterise the syntax used in Lisp. A comprehensive study of CLOS can be found in Steele (1990) and Keene (1989).

Lisp expressions, referred to as forms, are written so that the innermost forms are evaluated first and the outermost last. Forms are encompassed within a matching set of parentheses and denote a list. When a form is evaluated, its first element is expected to be a symbol that has a function definition. All remaining elements in the form are considered as arguments to the function, e.g., each of the following forms returns the value 6:

```
(+ 1 2 3)
```

```
(* 2 3)
```

```
(+ (- 4 2) (- 8 4))
```

```
(/ 12 2)
```

3.1.1 Concepts of Object-Oriented Languages

Several key concepts exist within an OOP language. Objects can be characterised as identifiable entities with a set of applicable operations and state, i.e., information can be stored into and retrieved from them. A set of abstraction techniques are used to minimise the number of details associated with a large number of similar objects, the most noteworthy of these abstractions being the concept of *class* (Awad, Kuusela & Ziegler, 1996).

A *class* determines the structure and behaviour of a set of objects, which are called its *instances*. In CLOS every object is an instance of a class; classes are themselves instances of meta-classes. Each instance is aware of its class and may exhibit structure in the form of a set of *slots* that act as local storage variables giving it a sense of state. The set of operations that can be performed on an object is determined by its class. A function whose behaviour depends on the classes of the arguments supplied to it is called a *generic function*. Each generic function contains a set of *methods* that define its class-specific behaviour and operations—a method is said to *specialise* a generic function. When a generic function is invoked, a subset of its methods based on the classes of its arguments is executed (Steele, 1990).

Classes can inherit structure and behaviour, i.e., slots and methods, from other classes that are called its *superclasses*. Proper utilisation of inheritance in OOP can make code reusable. Figure 1 demonstrates graphically the key concepts of class, inheritance, instance, function, and method. It also shows five examples of operations using the three topmost visible instances as arguments to the three generic functions.

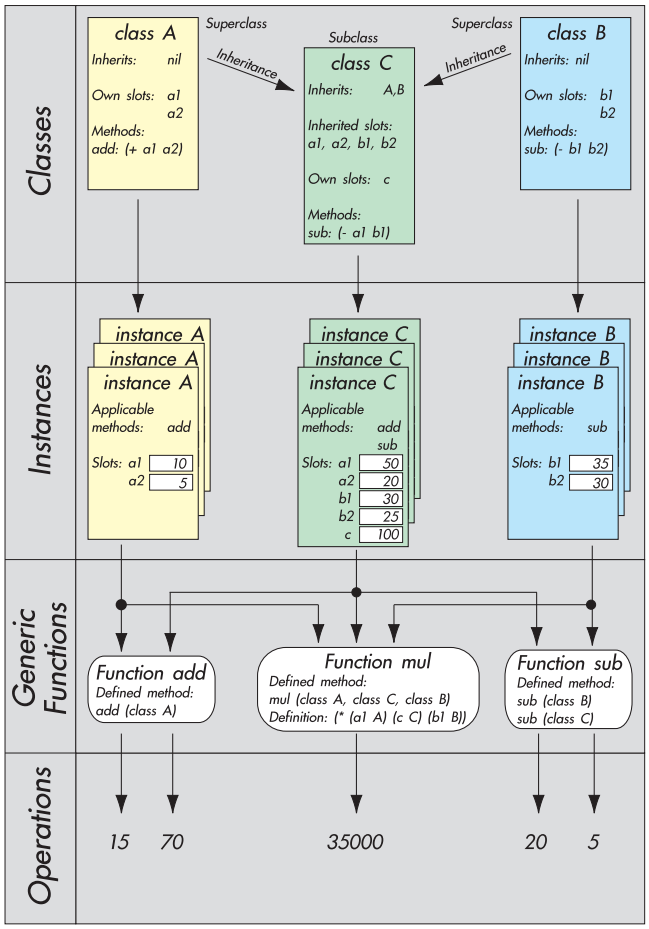


Fig. 1. Key concepts of object-oriented methodology.

For example, assume that class *A* is defined as well as a generic function called *add*. A method is defined for generic function *add* requiring a class *A* instance, or one inheriting from class *A*. When applied, this method performs addition on the slots *a1* and *a2*, the internal structure of an instance of class *A*. Now, if class *C* inherits from class *A* then *add* is also applicable to instances of class *C*. Likewise, class *B* can be defined, as well as a generic function called *sub*. A method for *sub* can be defined which calculates the difference between slots *b1* and *b2*. If class *C* also inherits from class *B* then an instance of class *C* will have 5 slots: *a1*, *a2*, *b1*, *b2*, and *c*, the first four slot definitions being inherited. Class *C* can define an additional method for *sub* thus specialising the behaviour of the generic function, i.e., the behaviour of *sub* will differ for instances of class *B* and class *C*. Finally, other generic functions such as *mul* can be defined and methods written for it that require several instances of different classes presented in a specific order to the function.

Instances of classes are created by the function `make-instance` that accepts a symbol argument specifying the class. For example, when evaluated, the form

```
(make-instance 'A)
```

returns a newly created instance of class *A*. Utilised extensively later in this thesis is the ability of an instance to test its class membership. The Lisp function `typep` (short for type-predicate) takes two arguments: an object to be tested and a symbol specifying a class. Function `typep` checks if the object's class inheritance tree includes the class specified and returns a Boolean value: `T` (for true), or `NIL`. For example, using the class hierarchy found in Fig. 1 the following forms return these values when evaluated:

```
(typep (make-instance 'A) 'A) ⇒ T
```

```
(typep (make-instance 'A) 'C) ⇒ NIL
```

```
(typep (make-instance 'C) 'A) ⇒ T
```

```
(typep (make-instance 'C) 'C) ⇒ T
```

In general, OOP languages do not possess built-in infrastructures for representing structural relations between objects that are available in some higher-level artificial intelligence languages. Relationships of the type one-to-one and one-to-many must be defined explicitly. For example, if the signals and speakers existing in a speech database are modelled as objects then it would be beneficial to link these objects to one another. In this way a speaker would know all of its utterances (one-to-many link) and each signal would know who its speaker is (one-to-one link).

In summary, the OOP methodology allows systems to be designed and problems to be modelled from smaller and tested software modules. Subsystems, e.g., in the form of classes, should strive to model their functions and inherent knowledge as orthogonally and independently as possible from other classes. The level of knowledge orthogonality achieved in OOP code determines largely how reusable it is in other applications.

3.2 Signal Representation and Processing Systems

Functional closures have been used as abstract program representations in programming language theory and as concrete program representations in languages such as Lisp (Kopec, 1980). When extended conceptually to the digital signal processing domain, a *closure model* represents a signal as parametrised procedures along with a set of parameter values. Representing discrete-time signals as

formal data abstractions was first explored in the late 1970's for filter design purposes (Oppenheim & Hamid, 1992). In the mid 1980's Lisp-based signal processing languages SRL (Kopec, 1985) and SPLICE (Myers, 1986) were developed that followed the *closure model* for signals. A speech analysis software system called SPIRE (Zue *et al*., 1986) running on Symbolics Lisp Machines was used extensively in many tasks such as in the development of the TIMIT corpus (Garofolo *et al*., 1993).

Popular commercial signal processing systems include MATLAB (The Mathworks, 2001) and Mathematica (Wolfram, 2001). The former has obtained wide support and is largely based on matrix operations to carry out signal processing efficiently. The latter is more adept for symbolic mathematical experiments. However, both systems are not well suited for working with or for modelling speech existing in large corpora since they are controlled by script languages. Also, creating specific types of data structures within their respective modelling languages is not efficient.

QuickSig—the system on which the work was performed—is described in more detail in section 3.5. It is a system which is motivated by the closure model with some compromises made for efficiency reasons.

3.3 Representation of Phonetic and Linguistic Information

Linguistics deals with the nature and structure of human speech—either in written or spoken form. Many theories and models exist within the field of linguistics which attempt to describe the complex relationships that exist, e.g., between words, phrases, sentences, and paragraphs. More specifically, *phonetics* is a branch of linguistics that focuses on the sounds of speech which includes their production, combination, description, and representation by written symbols. A written symbol representing a speech sound can be an element in a *phonetic alphabet*.

Representing information in a model from the field of phonetics and linguistics requires that relations and structures can be formed between objects such as words, syllables, and phonemes. Since relations can be added to the OO paradigm efficiently, it presents a good choice for an implementation to be based on.

Logic programming provides another alternative to processing phonetic and linguistic information. It is useful for programming logical processes and making deductions automatically. PROLOG, developed by Colmerauer and Roussel in 1971 is one of the most generally used logic programming languages. Designed initially for natural-language processing (Jurafsky & Martin, 2000) it has become a widely used language for artificial intelligence research.

PROLOG is suitable for problems in which logic is intimately involved, or whose solutions have a succinct logical characterisation. Like other interactive, symbolic languages, PROLOG can be used for rapid prototyping. However, representing structured data can be more naturally achieved with the object-oriented paradigm. Since Lisp/CLOS combines object-oriented, functional, and procedural programming within a single, unified system, it was chosen as the system implementation language.

3.4 Database Techniques

Much effort has been expended by the speech community in designing and producing speech corpora and databases. However, considerably less thought has been given to the type of database model that best represents speech for optimal database access. Most speech processing systems rely on some form of a structured database access mechanism. Since the relational database management system (RDBMS) is a well-established paradigm widely used in many non-speech fields it has been adopted to model spoken language as well. This section reviews an established database model, the RDBMS, as well as an emerging one, the object-oriented database management system (OO-DBMS).

3.4.1 Relational Database Management Systems

Relational databases are centred on the idea of two-dimensional tables which are called relations. Relational databases comprise multiple tables in each of which the values of a number of data keys are defined for each data entry. These tables are isomorphic to mathematical relations which have a solid theoretical foundation. Therefore, relational databases are well understood and are also technologically mature and stable. Queries in an RDBMS environment can be performed using a string matching formalism comparing the values of selected data keys within and across tables. Standard Query Language (SQL) is one such language which can be used to access one or more of the relational tables.

However, there are certain data structures that either don't fit well into relations; or that, when shoehorned into relations, do not lend themselves well to querying (Stajano, 1998). Analysis of spoken language often entails complex linguistic structures or objects that cannot be easily handled in a RDBMS, e.g., signals, arrays or lists of other objects.

3.4.2 Object-Oriented Database Management Systems

The OO-DBMS, more recently developed and less known than RDBMS, is also able to represent speech data but in a more natural manner. In an object-oriented environment data are modelled as objects; an OO-DBMS stores objects as well as their relationships with other objects. For example, objects such as speech signals, speakers, recording environments, equipment, annotations, etc., when selectively linked with one another create a structure where database access can be accomplished by

structure traversing search functions revealing desired contexts.

Object-oriented databases lack the mathematical foundation that relational databases possess and are not as well understood theoretically. Nevertheless, they support user-defined, arbitrarily complex, data types and can also deal with nested objects thus permitting efficient recursive queries. Of special importance is the characteristic that an OO-DBMS explicitly preserves type information across the database and application interface.

3.4.3 Impedance Mismatch

The preservation of strong typing between the application and the database eliminates the so-called *impedance mismatch* caused by the narrow and limited interface between a RDBMS and the application (Stajano, 1998). In an OO-DBMS the application and the database can be brought closer to each other and the programming language of the application can be integrated with the database's data manipulation language.

From a database processing viewpoint, speech data is by nature both unstructured binary signal data and highly structured symbolic data. The relational model supports only atomic and symbolic attributes organised in relational tables. It is thus not able to naturally model neither signal nor symbolic speech data. This is the fundamental reason for the impedance mismatch of the relational data model with respect to speech data (Draxler, 2001).

Effective representation of speech corpora requires structured models on which spoken language can be modelled. On the other hand, efficient utilisation requires that the speech model be well tuned and tightly coupled to the application which is using the database. The efficiency of the transfer of data between the database and querying application can be measured and is related to the impedance match between the two systems.

In general, an effective representation of speech is a prerequisite for efficient database access. However, a clear and structured model where speech information and data can be placed and relations drawn out explicitly does not guarantee that data access will be efficient. Impoverished representations of the results of database queries hinder the rate at which applications can access corpora. For example, queries returning only a textual description of a speech context in an indirect manner such as a file name representing a signal, a pair of indices representing the location of a phone, etc., exhibit a large impedance mismatch if further processing is to take place with other parts of the model structure. Every transaction between the database and application requires a wasteful transformation of representational form.

In electrical engineering, impedance is defined as a measure of the total opposition to current flow in an alternating current circuit. Made up of two components, ohmic resistance and reactance, impedance is represented as a complex value. In general, an analogy can be drawn from the physical realm to the software realm for defining the impedance of software modules. Whenever two software systems communicate with each other a transformation of representation may be required. This is wasteful since work must be expended to map from one representation, protocol, etc. to another. Impedance mismatch occurs in databases when a query language such as SQL is embedded in a programming language and the objects that are manipulated by the query language statements are not subject to the type checking constraints of the language (Hughes, 1991). To ensure the most efficient transfer of information the impedances of the communicating systems should be identical.

Figure 2 shows a series of software modules used to communicate with a speech corpus. At the left is the corpus in its media-distributed form. The next level is a database access interface to the speech corpus which provides a mechanism to retrieve data. The database access level may provide a level of abstraction that facilitates usage, e.g., instead of referring to file names in the corpus, an object-oriented representation may exist so that pointers to objects can be used instead.

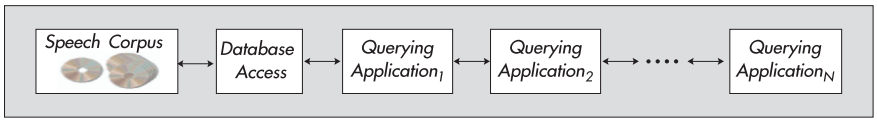


Fig. 2. Efficiency is dependent upon the individual communication efficiencies between querying applications, the database access system, and the speech corpus.

The remaining modules are a sequence of one or more querying applications that communicate with the database access module. These modules are used to access data and process the symbolic and numerical information, which at the highest level produce results that the user can process. The user may be an actual human or another higher-level software module that has a specific goal in mind, e.g., an artificial neural network whose task is to learn some pattern from a corpus. Between each querying application there may exist a necessity to change the representation form of the data, e.g., at a low level (near the speech corpus) a set of indices may be used to represent the sample values in a signal at some point in time. However, at the querying application level a complex, high-level object may be represented instead, e.g., a human uttering a sentence. Mapping between different representations during database access reduces the efficiency of the system—to reduce impedance mismatch a system should strive to use the same representation throughout all processing levels. Therefore, if high-level, complex data structures are used in the final querying application then the use of an object-oriented model throughout all layers is advocated.

3.5 The QuickSig Environment

QuickSig (Karjalainen, Altosaar & Alku, 1988; Karjalainen, 1990), a research-oriented digital signal processing environment, is motivated by the closure model for signals but is biased towards an engineering approach so as to improve computational efficiency. As in SPLICE, QuickSig is able to represent infinitely long signals and promotes signal domain concepts such as *support* and *interval* by explicit means provided within its signal representation language. From observations of the mathematics of discrete-time signals and their notation, five criteria can be used to measure the adherence of a signal processing system to the underlying mathematics (Oppenheim & Hamid, 1992). The external observable behaviour of QuickSig signal data objects meets the criteria of:

Criterion	*Description*
uniform reference	*Allows inquiring a signal's dimensions and sample values.*
dimensional extensibility	*The abstractions for one– and multidimensional signals are similar.*
immutability	*The observable properties of a signal object are defined when the object is first created and remain constant.*

QuickSig relaxes the latter condition of immutability for efficiency reasons by permitting signals to evolve over time. The remaining two criteria deal with how signals and systems are specified and are also supported in QuickSig:

Criterion	*Description*
manifest typing	*The ability to record history-based operations, e.g., information on how a signal was generated, thus permitting symbolic signal processing.*
deferred evaluation	*The ability to create a signal without computing its samples. QuickSig extends this criterion by allowing the loading of signal objects from its persistent object-oriented database without necessarily loading in sample values.*

Signals in QuickSig are objects with local variables that model the samples of a signal in some *support area* in some domain (Oppenheim & Hamid, 1992). Historically, *support area* was referred to as the

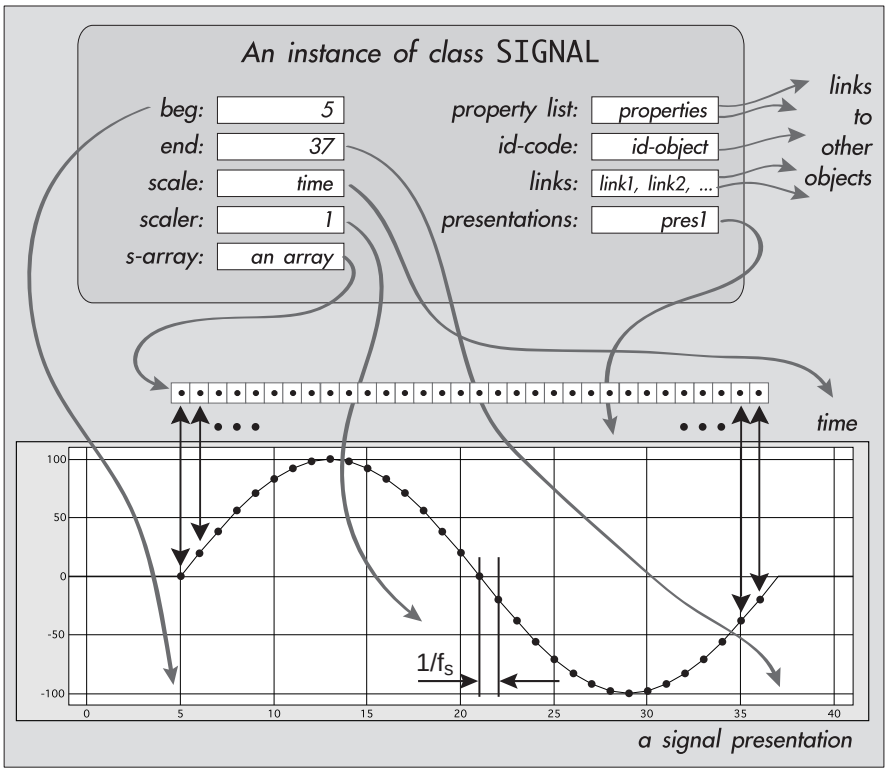


Fig. 3. An instance of class `SIGNAL` showing some of the object's slots and links.

non-zero extent region of a signal (Kopec, 1980) even though in QuickSig the value need not be zero when reading a sample value from outside of the signal's explicitly defined area. When writing a new sample value to a signal outside of its support area, the signal's sample array(s) are automatically adjusted and shifted, if required. This inherent support area processing is part of the infrastructure of QuickSig and frees the user and higher level applications from low-level bookkeeping. Figure 3 shows an instance of class `SIGNAL` including some of the slots and links it contains. Shown in the bottom part of the figure is a graphical presentation (also an object) of the signal where its `BEG` and `END` locations (in sample space) are defined, its `SCALE` (domain), `SCALER` (sampling interval), as well as its `S¬ARRAY` (sample array) are depicted. Some of the other slots that `SIGNAL` objects possess are also shown, e.g., a `PROPERTY-LIST` where any type of less frequently used properties or links of the object may be stored, an `ID-CODE` for inclusion in an object database, `LINKS` for defining relationships with other objects, and a list of graphical presentations the signal is involved with. For example, the latter is used for updating displays when a signal's value or property changes.

3.5.1 Signal Inheritance in QuickSig

In QuickSig a class hierarchy is defined that models different types of signals taking into consideration the characteristics of their samples. This is modelled generally in a similar way as the Lisp class `NUMBER` and its sub-classes are defined, see Fig. 4.

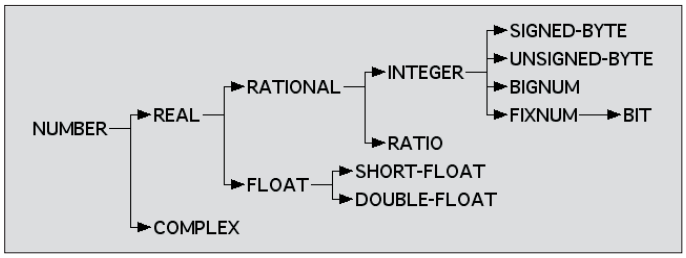


Fig. 4. Class inheritance tree for the Lisp class `NUMBER`.

In QuickSig the class `SIGNAL` specifies that its instances are allocated a sample array which can store any type of object, e.g., an integer or a floating point number, or any other more complex object such as another signal, etc. More specific classes based on `SIGNAL`, such as `INTEGER-SIGNAL`, limit their sample values to a certain data type and range. These class specialisations are used for both execution speed and storage optimisation purposes. Figure 5 shows the evolutional class inheritance tree for class `SIGNAL`. On the left-hand side are classes that contribute to the `SIGNAL` class with structure and methods. On the right-hand side are classes that inherit from class `SIGNAL` and represent specialisations.

3.5.2 Signal Processing Methods in QuickSig

Signal objects can be created in QuickSig in a number of ways: by manually specifying a set of sample values, by supplying a user-defined function, or calling an already existing function defined in a library. For example, to create a new sinusoidal signal object the function `make-sync-sine-signal` can be called with additional arguments specifying the amplitude, frequency, and number of desired samples. The following form creates a new signal and binds it to the variable `sig1`:

```
(defparameter sig1
   (make-sync-sine-signal 100 100 :size 512))
```

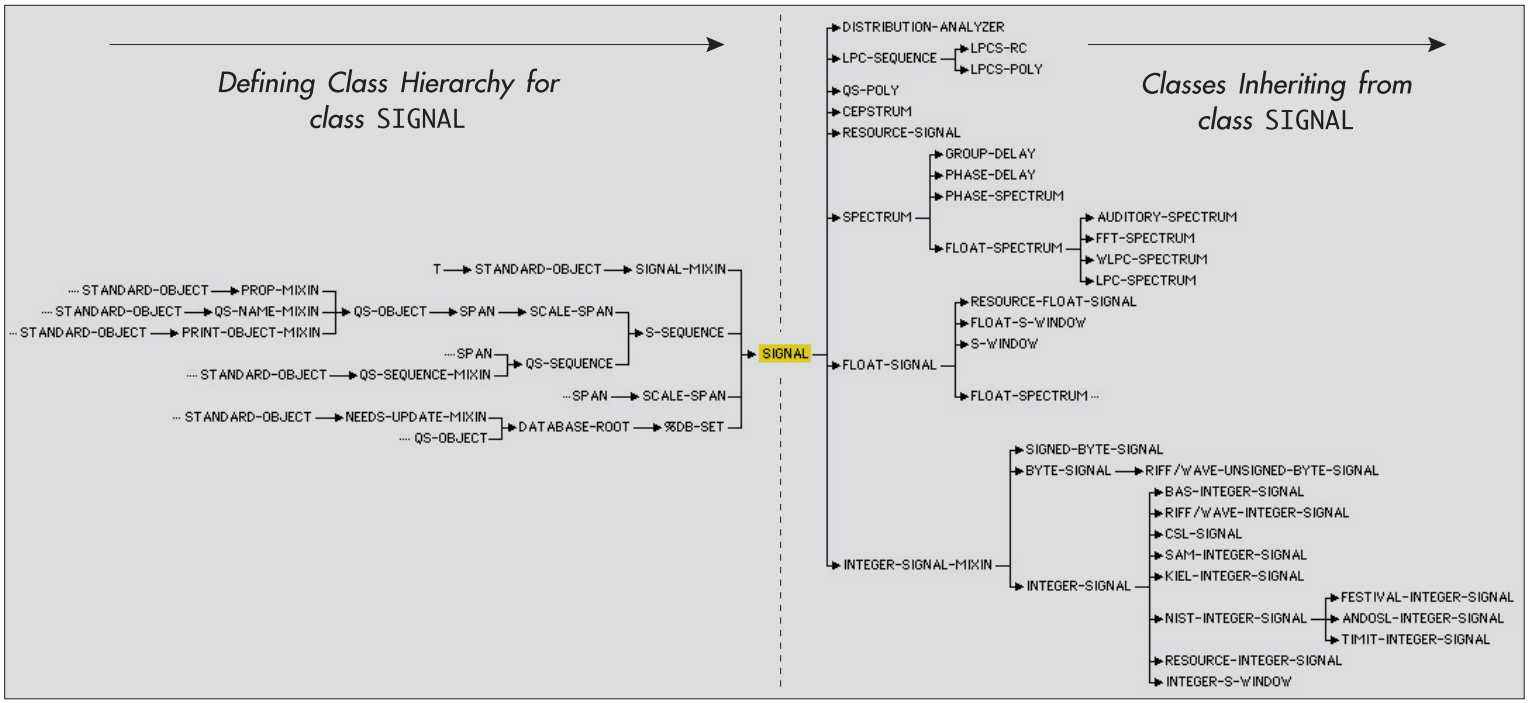


Fig. 5. Evolutional class inheritance tree for the QuickSig class `SIGNAL`.

Signals have generic signal processing methods defined for them, such as *add*, *sub*, *mul*, and *convolve*, and can operate on one another to form new signals, e.g., the Lisp form:

```
(add sig1 sig2)
```

combines the samples of *sig1* with *sig2* and returns a new signal object that represents their summation. The function `qs-call` is used to define such functions, e.g., `add` could be defined by the form:

```
(defmethod add ((s1 signal) (s2 signal))
   (qs-call '+ s1 s2))
```

Here the Lisp function `+` is passed as an argument to `qs-call` indicating that addition is to be performed on a sample to sample basis, along with two signal objects `s1` and `s2`. The function `qs-call` can be used directly as well to define any desired operation. For example, if the integral sum of the addition of two existing signals, `sig1` and `sig2`, needs to be computed, the following form can be evaluated to define a new signal called `integral-sig`:

```
(defparameter integral-sig
  (let ((sum 0))
    (qs-call #'(lambda (x y) (incf sum (+ x y)))
             sig1 sig2)))
```

In the above form the local variable `sum` is first bound to the value 0 and has its value updated within the lambda form during the summation process. Finally, signals can be visualised in QuickSig, e.g., the form:

```
(show (list sig1 sig2 (add sig1 sig2)
            (mul sig1 sig2) (convolve sig1 sig2)
            integral-sig))
```

results in Fig. 6 being drawn. Here two sinusoidal source signals, `sig1` and `sig2`, as well as their addition, multiplication, convolution, and integral sum, as defined in the above form, are shown. Dashed lines indicate areas outside of a signal's support area.

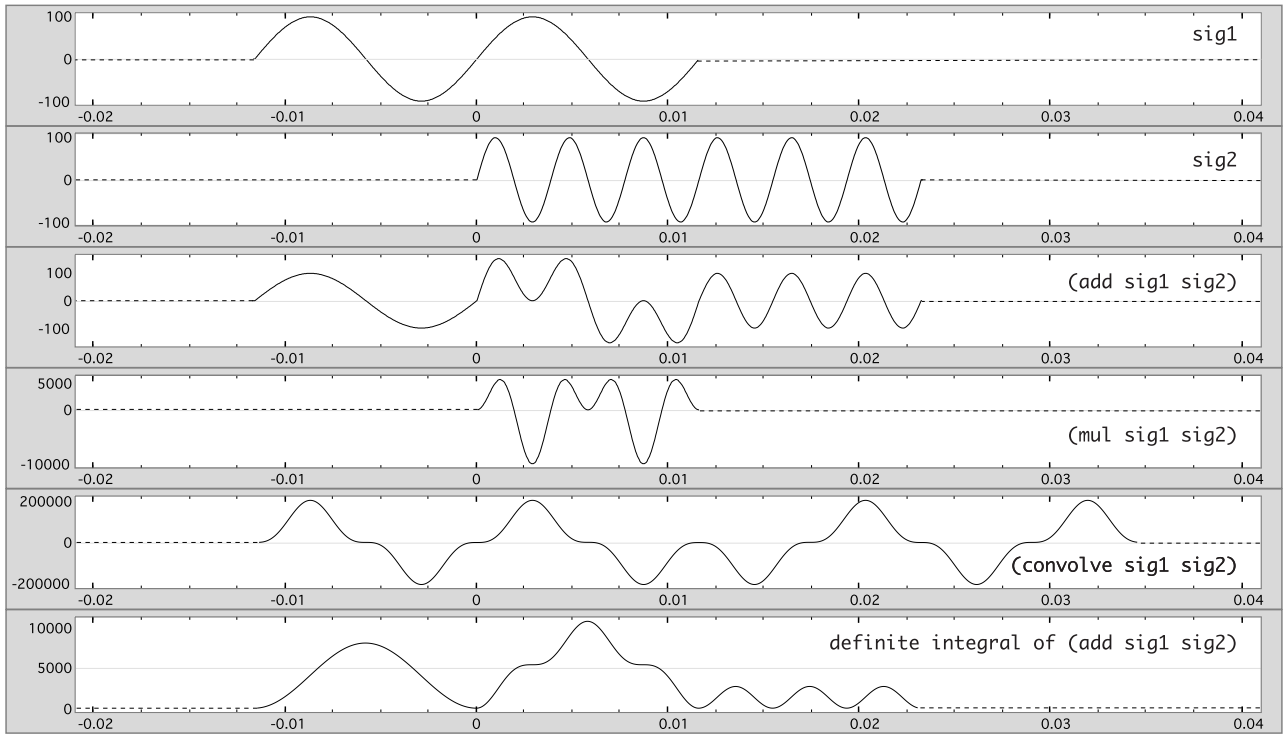


Fig. 6. Graphical depiction in QuickSig of two source signals `sig1` and `sig2`, as well as their addition, multiplication, convolution, and definite integral of their sum.

3.5.3 Modelling in QuickSig

QuickSig is used to model objects other than signals such as annotations, speakers, recording environments, and entire speech corpora. Persistent objects are supported by allowing objects to survive beyond the execution of the creating process. This is accomplished in QuickSig through an OO-DBMS that transfers objects to and from permanent storage transparently without user interaction allowing the objects to be reused in other processes and sessions.

QuickSig has been applied foremost to speech applications such as analysis, synthesis, recognition, speaker verification and identification (Altosaar *et al.*, 1998; Karjalainen, Altosaar, Alku, Lehtinen & Helle, 1989; Altosaar & Karjalainen, 1989; Karjalainen, Altosaar &Vainio, 1998) but also to other problem areas that involve signal processing, e.g., computer music (Karjalainen, Välimäki, Altosaar & Helle, 1992). The wide spectrum of different applications has shaped QuickSig into becoming a domain-independent signal processing environment which aims towards forming generic data abstractions and related methods—promoting the concept of knowledge orthogonality referred to in section 3.1.1. QuickSig provides an extensive multi-disciplinary infrastructure on which speech can be modelled. The interactive environment of the Lisp language is an ideal setting for the exploration and modelling of annotation data. Combined with object-oriented programming, QuickSig permits a high level of expressive freedom that may be difficult to obtain with other non-object-oriented speech processing systems. The remainder of this thesis describes the representation framework developed for corpus and database speech in QSSDB, the QuickSig speech database system.

4 Representation Framework

4.1 Nomenclature and Purpose

Speech corpora contain collections of data, i.e., information, that can be transformed into knowledge to further improve the understanding of spoken and written natural language (Altosaar & Vainio, 1999). The effectiveness of this transformation is dependent upon the quality of the data representation that is formed. An inadequate representation will limit the transformation and allow only a fraction of a speech corpus' potential inferences to be formed. Before arguing the purpose for a representation framework, some terminology local to this thesis is defined.

4.1.1 Nomenclature for Representations

Terminological discrepancies in speech and language technologies have caused misunderstandings between engineers, phoneticians, and linguists (Barry & Fourcin, 1992). Frequently used terms such as "level", "tier", and "domain" have different meanings depending on the school of thought and thus are used to mean different things. The emergence of new areas of study within human communication, such as audio-visual speech, have brought along their own terminology complicating the issue even further. Therefore, with due respect to previous work performed at defining a uniform terminology, a set of terms that strive to be theory-neutral and generic are defined here.

- The representation of a message that a human produces, measured by all or some of the channels it is communicated by, e.g., acoustic, visual, physical, etc., exists within a *representation framework* (abbreviated as *framework).* Specifically, each utterance in a corpus is modelled by a *framework*. Unique *frameworks* can be linked to one another to enlarge context, e.g., in dialogues.

- Information may be represented in different domains, e.g., a signal can be represented independently both temporally and spectrally. The signal measured within a communication channel, e.g., sound-pressure, light intensity, mechanical pressure, etc., including any of its possible derivations, is defined to exist within a *representation domain* (abbreviated as *domain).* Theories and representations of natural language, e.g., phonetics, linguistics, prosody, orthography, etc., have unique *domains* defined for them.

- Each representation domain may have one of more distinct *representation levels* (abbreviated as *level)* that define the structural hierarchy across different scales or resolutions. For example, the linguistic domain may contain the following *levels* among others: sentences, words, syllables, and phonemes. In some domains, *levels* do not necessarily have to follow a hierarchical or dominance

ordering.

- Each representation level contains a set of similar *representation units* (abbreviated as *unit)*, e.g., phones, visemes, etc. *Units* represent the actual data existing within a speech corpus.

- Representation domains can be envisioned as planes that are aligned along the temporal axis. However, the information stored in them need not be chronologically ordered since representation unit locations in a level can be expressed freely by *links*, e.g., in semantics where interpretation is possible only when an entire utterance has been processed. *Links* are used to indicate relationships between *domains*, *levels*, and *units*—thus enabling inferences to be generated.

4.1.2 Purpose

Numerous competing spoken and written natural language theories exist. A unique corpus can therefore be used to evaluate these competing theories when its utterances are modelled explicitly, as can be done with representation frameworks. Differences also exist in the constraints these theories place on the representational resources required to create models of speech annotations. Theoretically, a representation framework allows for the instantiation of one or more of these target theories in the form of a model. This not only allows theories to be compared by their common references (Barry & Fourcin, 1992), i.e., the quantifiable signals, but also promotes their further development and evolution in a common environment. Practically, a representation framework serves as a structure that binds together the different representation domains of an utterance much like a multi-layered blackboard. Information from different domains can then be analysed in a context-rich environment, i.e., in the presence of other related information. Inferences generated from analysing the representation units as well as the links between different domains, levels, and units can be used to generate new units, levels, and domains within the same framework. These new representations, possibly of a higher or lower abstraction, in the same or different domain, can be utilised in actual language applications, e.g., in speech synthesis where the transformation of linguistic words to phones via a phoneme level is accomplished, or, in speech recognition where the suitability of different word hypotheses in a semantic context as a function of time may be evaluated.

The representation domains, levels, and units that are mandatory within a framework depend upon the language-theory being modelled and which inferences are to be made and exercised by the application level. Requisite representation domains, levels and units are created by the user by defining new object classes and methods that encompass the knowledge required to support a certain theory.

4.2 Generic Representation of Data

Information existing within speech corpora needs to be evaluated in a specific environment for subsequent extraction of spoken language knowledge to take place. A priori knowledge of, e.g., file and directory formats, phonetic alphabets, labelling strategies, etc., is required for the correct interpretation of individual elements within files. However, this meta-knowledge specifying the location, format, type of each data item, and their relations to other data, is not always available in a computer readable format. This may force specific assumptions to be made at later and non-optimal processing stages, e.g., during queries. Immediate reinstatement of identity and meaning for information read from speech corpora promotes a reduction in the complexity of a generic modelling environment. For these reasons, a primary level of abstraction is developed in this work for annotations and signals existing in a corpus. This is accomplished through the use of the *corpus specification model* formalism.

4.2.1 Corpus Specification Model

Due to the wide variety of existing speech corpora standards, the need for a *corpus specification model* (CSM) is suggested. A CSM is an object which contains information regarding the location of data within a corpus, e.g., the file directory, name, extension structure, possible network address, protocol, etc., needed to retrieve information. CSMs specify the file formats encountered within a corpus, e.g., AIFF, WAV, NIST headers, annotation formats, etc., and in actual computational environments offer an interface to access data items in any desired order. By providing this interface, CSMs hide the intricacies of a specific file system from the user. CSMs provide, either algorithmically or through a lookup table, a symbolic description for each data item within a corpus. They offer a model in which hierarchical relationships between data types are defined. Finally, a CSM is able to provide a table of contents of its corpus with respect to different perspectives, a plan to access any or all of its corpus' data, and is aware of corpus errata, indicating means for remedy, either locally or externally through a network. Ideally, a CSM should be included with the corpus on the same distribution media or network. If a corpus is defined in an object-oriented database management system, where data, type, and relations are tightly integrated, this information could exist from the moment the corpus was created (Karjalainen & Altosaar, 1993). Since the speech corpora in Table 1 do not include this data explicitly, except for Finnbase due to its OO structure, CSMs have been defined and implemented in QSSDB for ANDOSL, Estonian, Kiel, and TIMIT.

4.2.2 Corpus Parser and Generic Representation Format

A single *corpus parser*, supplied with information from the corpus' CSM, is able to access all raw corpus data regardless of storage format. A parser reads in information from a corpus and produces

objects that are explicitly typed, i.e., extensive information about the retrieved data is packaged along with it. These new objects are instances of the *generic representation format* (GRF) class in which any type or amount of information can be stored. Parsed annotation data, as GRF objects, consist of the exact textual strings found in the corpus but are void of any line termination characters, data partition symbols, and anything else that does not represent the annotation itself. GRF objects representing phones thus contain associated information such as their respective phonetic alphabet, character set, label syntax, annotation style, language, and speaker, when appropriate, either locally or through a hierarchy of GRF objects. The latter is useful if a compact representation in terms of memory usage is required. If temporal information is available, a GRF object is assigned an *interval*, which represents its extent or location in time. In general, and if available in the corpus, *events* can be used to define any kind of an object marker in one or many domains, e.g., a time-frequency area indicating the temporal interval and spectral energy range of a fricative. GRF objects are linked sparsely according to the hierarchical model suggested by the CSM. For example, annotation GRF objects for an utterance are contained in ordered lists, one list for each different type of annotation supplied in the corpus. Any number of annotations may be linked to a single acoustic waveform, e.g., several parallel but differing annotations may exist for a signal when both manual and automatic segmentation and labelling has been performed. Data from corpora, in the form of GRF objects, can be presented to the user graphically for viewing and editing, as seen in Fig. 7. The graphical representations for GRF objects in this figure consist of a signal waveform and annotation levels covering the phone, the phonemic canonical word form, orthographic phrases, sentences, and paragraph levels. These GRF objects are produced by the corpus parser operating on a set of files representing an utterance found in the Kiel Corpus of Read Speech (`KIEL CORS:PH90:BUTTER:K22BUTT2`).

Figure 8 shows in more detail the annotations over a 300 ms section of speech. Highlighted in the graphical presentation is a GRF phone object, and when queried about some of its properties, it returns the following typed Lisp values (the variable `x` is bound to the selected object):

```
(symbolic-domain x)    ⇒ phonetics (a symbol)
(hierarchical-level x) ⇒ phone (a symbol)
(character-set x)      ⇒ German-ISO-8859-1 (a symbol)
(phonetic-alphabet x)  ⇒ German-SAMPA (a symbol)
(annotation-syntax x)  ⇒ Kiel-1 (a symbol)
(interval x)           ⇒ [16.389, 16.466] time in seconds (an interval object)
(item-value x)         ⇒ "##Q- $a- $l- $s-+ ##S- $o:- $n-+  ##m" (a string)
```

The GRF formalism allows for flexibility in representing annotations. For example, adjacent phones in the same utterance may be annotated using different phonetic alphabets, or, each word, phrase, sentence, etc., may be from a different language. In addition, units such as words may have associated with them a unique speaker—as is the case when multiple speakers exist within one or more signals—by linking the word object to a speaker GRF object. In typical situations the wealth of information supplied by GRF objects may not all be required concurrently by the user or system. However, the availability of this information in a generic structure eases the development of the second level of abstraction for annotation data described in section 4.3.

The first three top left panes in Fig. 9 show corpora, a set of CSMs, and the corpus parser, for generating and viewing GRF objects. The other panes in this figure are referred to in later sections.

4.3 Forming Representation Frameworks

The formalism described in the previous section provides a primary level of abstraction to corpus annotations. By reinstating missing type and relationship information, data is expressed in a generic manner. However, data, type, and relationships are still not sufficiently integrated to model speech units effectively and allow for efficient queries—they need to be fused with appropriate knowledge sources that will give depth and meaning to the speech units in several dimensions. Since GRF objects reflect to a large degree the spoken language theory subscribed to by the corpus developers, e.g., the style of phonetic segmentation employed, they may need to be reorganised according to a desired target theory, as mentioned in section 4.1.2. A *compiler* accepts GRF objects and produces representation units while a *linker* associates these units with one another to form the levels and domains of a representation framework; refer to the top-right pane in Fig. 9. Resources and techniques found useful in developing a second level of abstraction to corpus annotations are now described.

4.3.1 Resources

Compiling and linking require domain-specific knowledge which is supplied by a set of resources. This is desirable so that the compiler and linker can be void of domain-specific knowledge allowing them to operate generically. An important goal when designing resource objects is to make their knowledge content as orthogonal, with respect to other resources, as possible. This promotes their reusability with other corpora and minimises the writing of new code. Implemented as object classes and instances in QSSDB, resources are defined for distinctive features, phonetic alphabets, natural languages, character sets, hierarchical scale, and labelling syntaxes. Other resources, when required by a new corpus or target theory, can be added later incrementally.

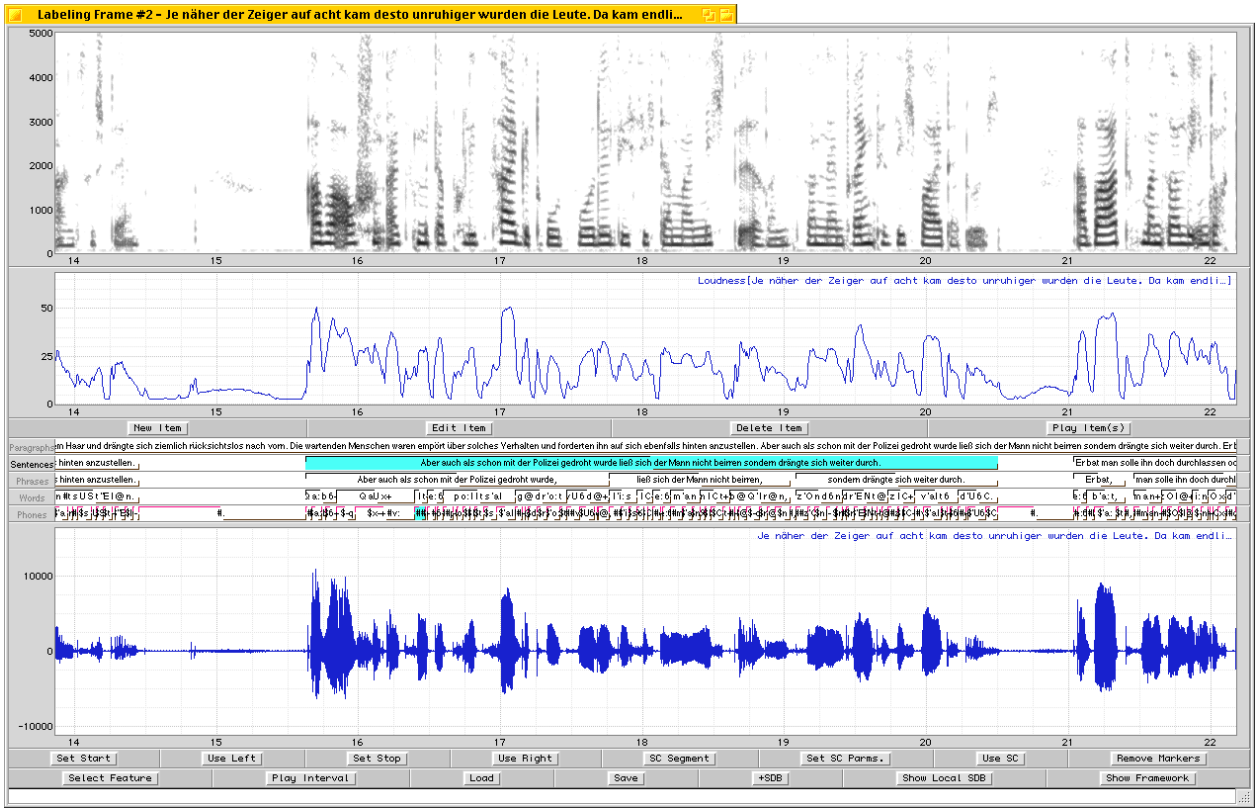


Fig. 7. Part of an utterance from the Kiel Corpus of Read Speech presented in a labelling frame. Highlighted sentence is: "*Aber auch als schon mit der Polizei gedroht wurde ließ sich der Mann nicht beirren sondern drängte sich weiter durch*."

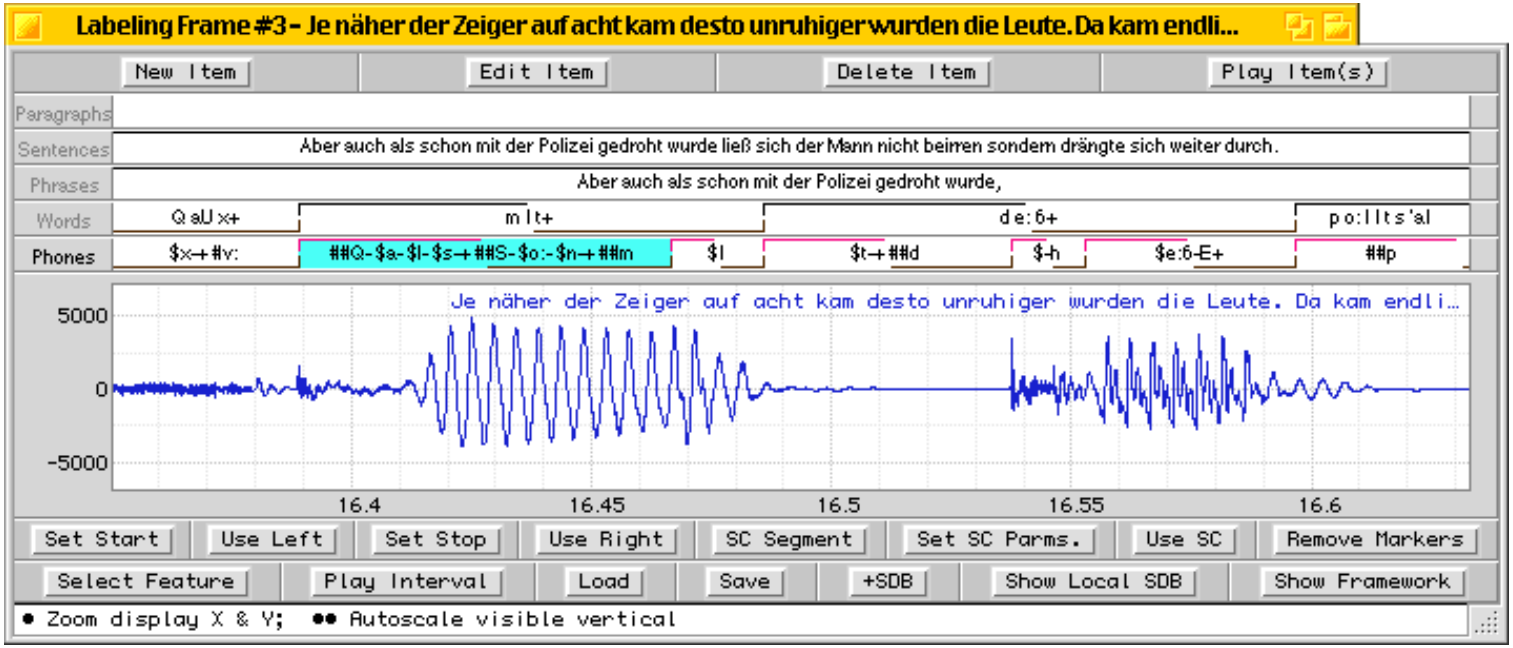


Fig. 8. GRF objects viewed along the time axis over a 300 ms section of Fig. 7. GRF objects encapsulate, in addition to their labelling string and time interval, other information such as their domain and hierarchical level, character set, phonetic alphabet, labelling syntax, speaker, and signal.

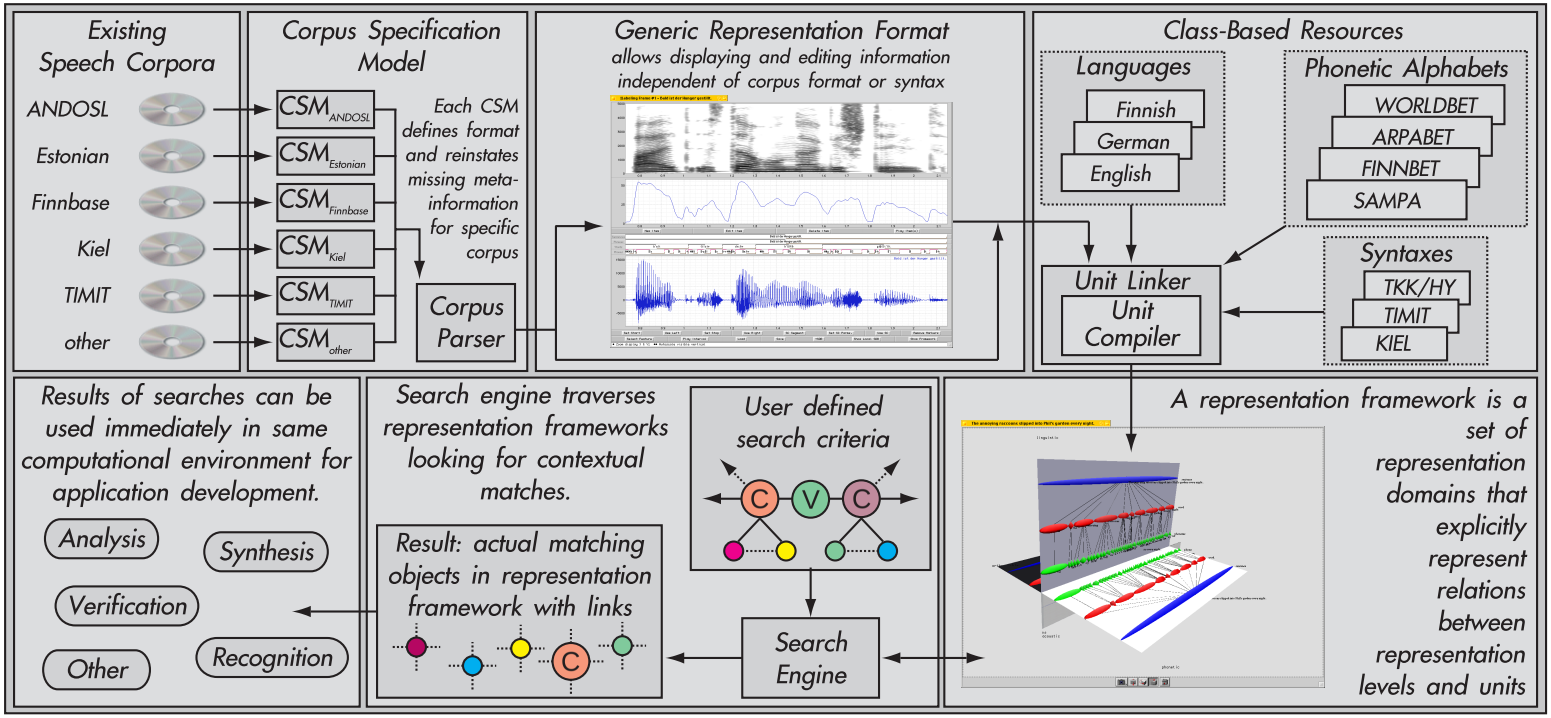


Fig. 9. Overview of transforming different speech corpora into representation frameworks of spoken language. Results of searches performed on these generic models can be used in different applications.

4.3.1.1 Worldbet and Distinctive Features

Numerous machine-readable phonetic alphabets have been defined and are usually specific to one language group sharing the same speech sounds. A corpus is therefore typically labelled in the language's "native" phonetic alphabet, e.g., German-SAMPA, ARPABET (American English), etc. However, even the same corpus may be labelled using different "foreign" phonetic alphabets, e.g., as is the case with Finnbase where Finnbet and Worldbet are employed. This variety of symbol sets may impede cross-corpus and cross-lingual studies unless a common phonetic alphabet is available with which any sound can be specified. Worldbet (Hieronymus, 1993) is one such phonetic alphabet and addresses the lack of a standard phonetic labelling system by providing a common formalism for representing the sounds existing in all of the world's languages. It is an ASCII version of the International Phonetic Alphabet (IPA) (International Phonetic Association, 1999) and has in addition broad phonetic symbols not presently in IPA.

An object-oriented implementation of Worldbet is supported in QSSDB. Figure 10 shows the root class *phonetic-feature* as well as the hierarchy of its sub-classes (Steele, 1990) where some of the nodes represent distinctive features (Chomsky & Halle, 1968). To model a unique sound in the set of all human sounds, single leaf nodes are chosen from some of the appropriate terminal sub-trees and are used to define a new sound class. Instances of these classes form representation units such as specific phones and phonemes.

For example, the near-open, front, unrounded, voiced monophthong phone [æ] (IPA), has its representative class inheritance tree drawn in reverse order in Fig. 11. Here it can be seen which classes comprise the phone; the classes involved can be used later in forming database queries. The phone class [æ] (IPA) inherits the `NEAR-OPEN-FRONT-UNROUNDED-VOICED-MONOPHTHONG` class, which itself inherits the feature classes highlighted in Fig. 10. Also inherited is the class `PHONETIC-PHONE-UNIT`, which adds further characteristics to instances of class [æ] (IPA) giving them knowledge about their:

- domain, i.e., phonetics
- hierarchical order in a scale of structural units defined for this phonetics theory, e.g., … word, syllable, phone, segment, ...
- data nature, i.e., symbolic rather than numeric
- temporal interval
- available access mechanisms through links to other units in the framework

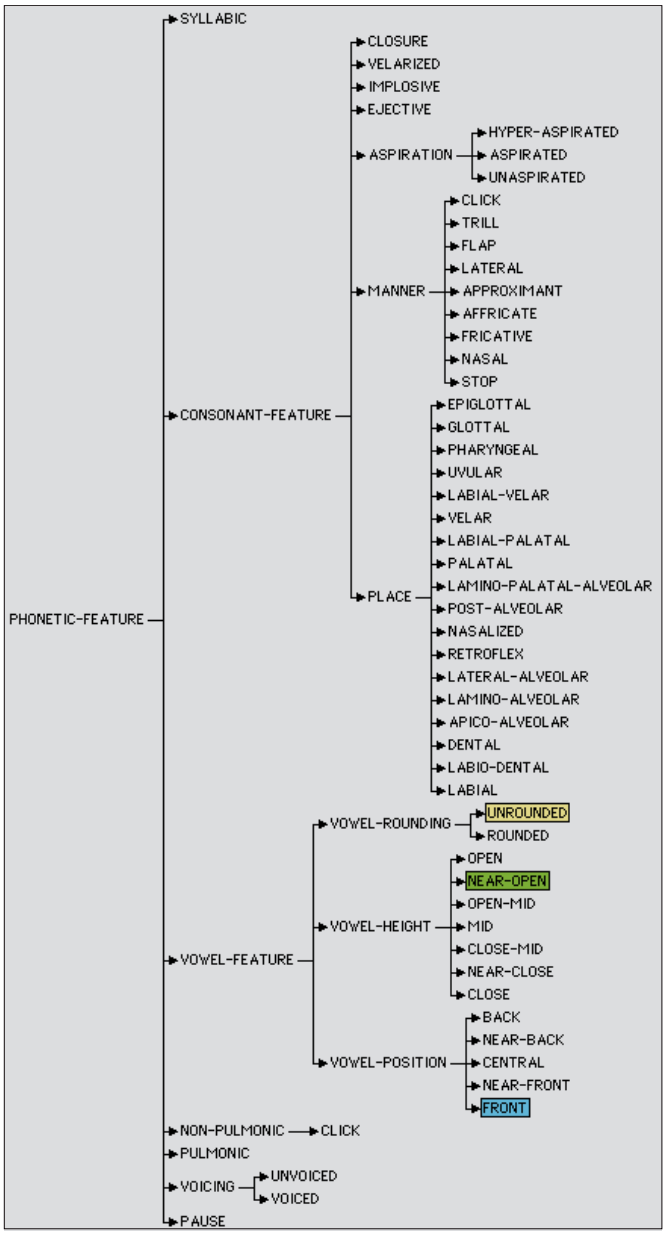


Fig. 10. Class inheritance tree shown from class `phonetic-feature` downwards. The three highlighted classes are used to define the prototype phone/phoneme [æ] (IPA) class.

Any instance of a class inheriting from class `phonetic-feature` can have its class membership tested using the function `typep`, as was described in section 3.1.1. If `x` is a variable whose value is an instance of class `NEAR-OPEN-FRONT-UNROUNDED-VOICED-MONOPHTHONG-PHONE`, i.e., an [æ] (IPA) unit, then `x` can be efficiently tested for properties such as nasality, voicing, and whether `x` is a front

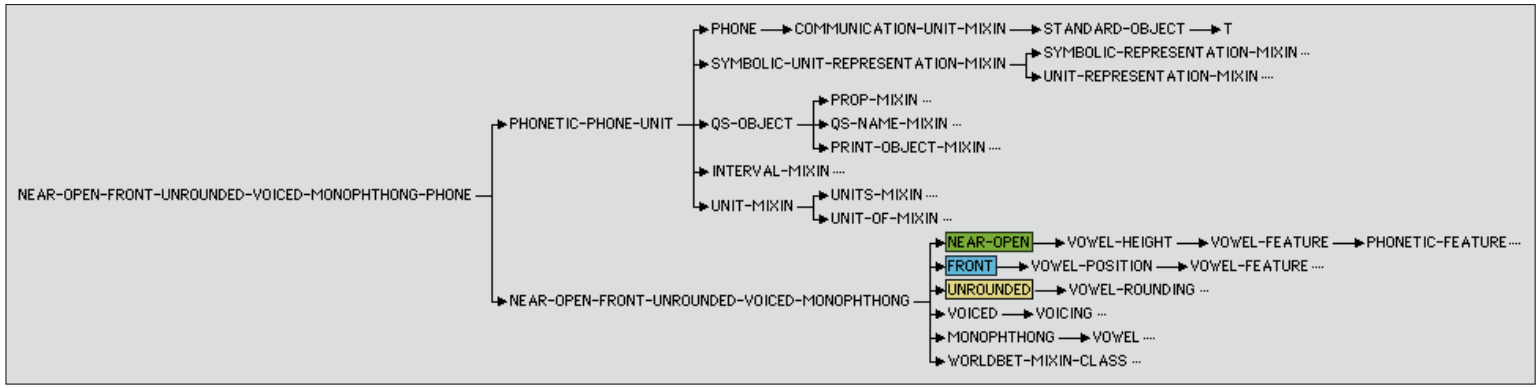


Fig. 11. Reverse class hierarchy tree for the class representing the prototype phone/phoneme [æ] (IPA).

vowel, by the following three forms, respectively:

```
(typep x 'nasal) ⇒ NIL
```

```
(typep x 'voiced) ⇒ T
```

```
(typep x 'front) ⇒ T
```

4.3.1.2 Phonetic Alphabets

Speech from a specific language or dialect utilises only a subset of all human sounds, referred to as a *sound set*. If every element in a sound set is associated with a unique symbol then this subset can be referred to as a phonetic alphabet. Assuming that similarly produced speech sounds can exist within different dialects and languages, Fig. 12 a) shows how each language selects from IPA, or from the larger Worldbet set, the needed sounds to adequately represent its own sound set. Figure 12 b) shows the different symbols associated with the same sound, e.g., [æ] (IPA). In QSSDB, over 200 prototype Worldbet classes are currently defined from which a similar number of phone and phoneme classes are specified. Six phonetic alphabets, referred to in Table 1, are currently defined. A macro expression that converts a phonetic or phonemic label string to a symbol that represents a Worldbet class, is available for each phonetic alphabet. For example, the following forms all evaluate to the same symbol (Australian English and German do not include this sound in their sound set):

```
(Worldbet "@")  ⇒  NEAR-OPEN-FRONT-UNROUNDED-VOICED-MONOPHTHONG
```

```
(Finnbet "ä")   ⇒  NEAR-OPEN-FRONT-UNROUNDED-VOICED-MONOPHTHONG
```

```
(ARPABET "ae")  ⇒  NEAR-OPEN-FRONT-UNROUNDED-VOICED-MONOPHTHONG
```

Since these macros execute at compile-time rather than at run-time, database queries do not deal with string comparisons but rather with symbols whose equivalence checking is considerably more efficient. Examples of database queries employing mixed machine-readable phonetic alphabets are given later on.

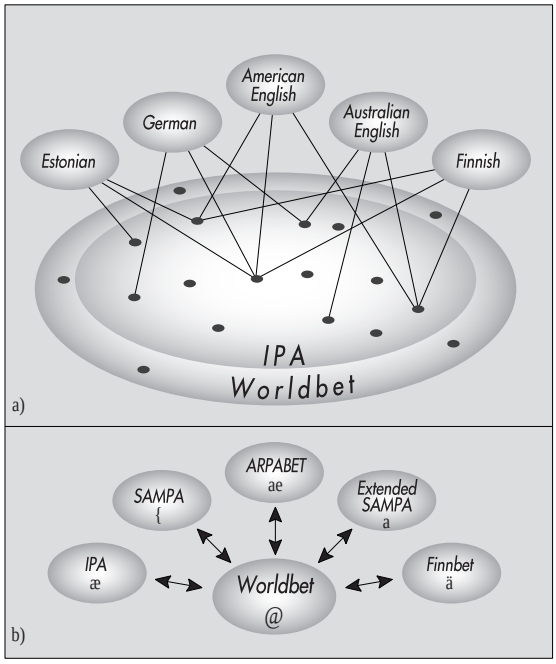


Fig. 12. a) The concept of sound sets for different languages. b) Symbols used in different phonetic alphabets to represent the same near-open, front, unrounded, voiced monophthong.

4.3.1.3 Natural Languages and Dialects

Languages such as Finnish, English, German, etc., are defined as classes. Each language may have dialectal sub-classes whose pronunciation deviates from the standard form by having associated with it a different sound set. The phonetic realisation of individual speakers is also modelled and has been found useful when training speech synthesisers with QSSDB. In this case training material consisting of speech from a single person is used. Character sets are also resources and define the orthographical symbols used in a language or corpus.

4.3.1.4 Hierarchical Scale of Structural Levels

Different speech units, ranging from temporally short segments, such as stop releases, to increasingly longer units such as phones, syllables, words, sentences, etc., may be placed on an abstract axis having discrete symbolic levels. This *hierarchical scale* is conceptual in nature and assigns only a structural order to its elements. For example, linguistics may have the following levels defined: *text, paragraph, sentence, phrase, word, compound-word, syllable, phoneme,* and *mora*, with the latter two existing on the same level. The different speech units which form the hierarchical scale can be used in differing representational domains, e.g., the concept of *word* can exist in a orthographic, linguistic as well as a phonetic sense. Therefore, elements of the hierarchical scale are defined as prototype

classes, e.g., `TEXT`, `PARAGRAPH`, etc., and these classes are inherited by both the *representation level* and *representation unit* classes. This propagates orthogonal knowledge regarding each element type.

4.3.1.5 Labelling Syntax

If the syntax of the annotation scheme for speech units is straightforward, a direct approach can be taken to interpret annotation strings, e.g., as is the case with orthographic words. However, if the description of speech is rich, e.g., Kiel annotations, an interpreter based on explicit state transitions is desirable. Figure 13[1] shows part of a state transition diagram that interprets Kiel-1 phonetic level annotations containing information related to several domains, e.g., phonetics, linguistics, etc. Also shown is the interpretation of the labelling string "`##Q- $a- $l- $s-+ ##S- $o:- $n-+ ##m`" which was exhibited in Fig. 8. For example, when the interpreter processes the first substring, "##Q-", it produces three observations:

1. "##" ⇒ beginning of word (a useful observation for linguistics, phonetics, orthography, etc., since these domains may exist in the representation framework).

2. "Q" ⇒ a phone string in German-SAMPA; the German-SAMPA phonetic alphabet resource reduces this to the class symbol `UNVOICED-UNASPIRATED-GLOTTAL-STOP-CONSONANT`.

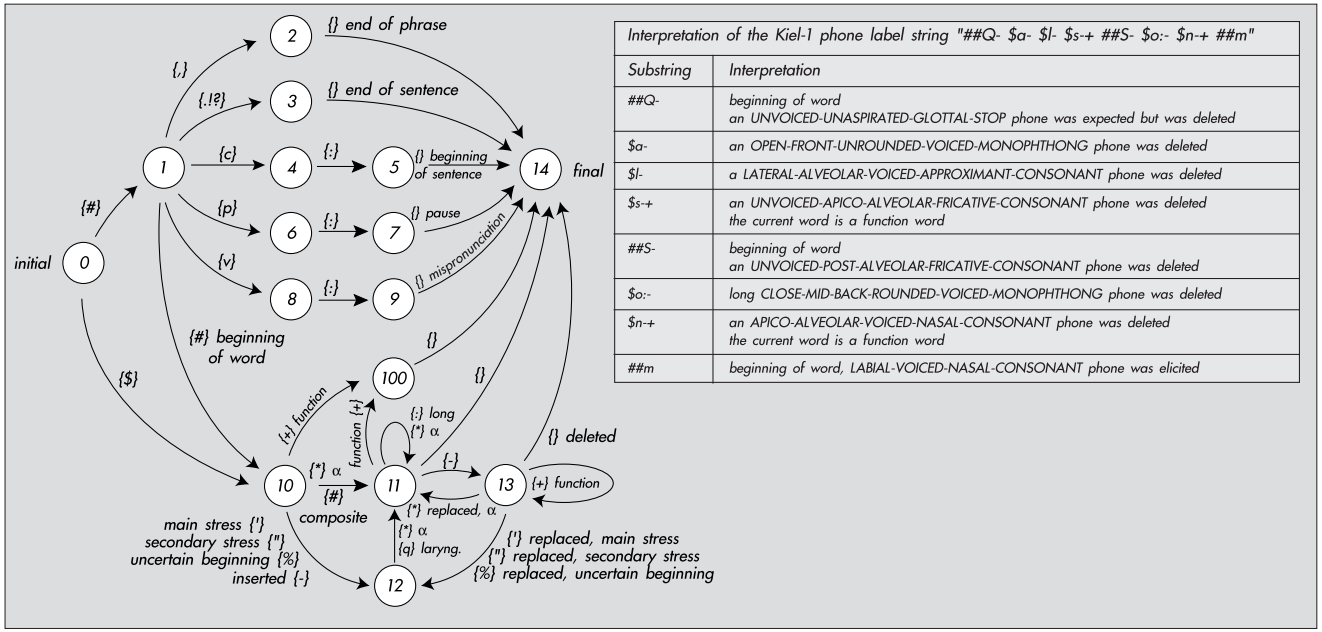
3. "-" ⇒ indicating that the phone was deleted by the speaker, i.e., the phone does not exist in the phonetic domain even though the canonical form expected it.

The above observations are posted to the domains they affect and then the following substring, in this case "`$a-`", is processed.

4.3.2 Modelling Local Properties

Since vowels are always presumed to be vocalic, they inherit the distinctive feature class `VOICED`, e.g., [æ] (IPA) in Fig. 11. If whispered, however, vowels are unvoiced. One way to model this is to locally store within a unit the deviating properties and to have the default characteristics defined by general class behaviour. If local property modelling is in effect for some aspect of a unit, search functions check for local overriding properties first, and if none are found in the unit, class behaviour is resorted to. This approach permits a wide degree of freedom in modelling individual variation in phones and other units while retaining storage efficiency since only exceptions are specified.

[1] The author along with Martti Vainio developed the state transition network required to interpret Kiel-1 phonetic level annotations. The automata evolved using a mainly trial-and-error method, along with helpful consultations from Klaus Kohler. The author of the thesis is responsible for the algorithmic implementation, Fig. 13 itself, and determining 68 cases where errors exist within the annotations of Kiel-1 annotations as found on the CD-ROM.



Interpretation of the Kiel-1 phone label string "##Q- $a- $l- $s-+ ##S- $o:- $n-+ ##m"

Substring	Interpretation
##Q-	beginning of word an UNVOICED-UNASPIRATED-GLOTTAL-STOP phone was expected but was deleted
$a-	an OPEN-FRONT-UNROUNDED-VOICED-MONOPHTHONG phone was deleted
$l-	a LATERAL-ALVEOLAR-VOICED-APPROXIMANT-CONSONANT phone was deleted
$s-+	an UNVOICED-APICO-ALVEOLAR-FRICATIVE-CONSONANT phone was deleted the current word is a function word
##S-	beginning of word an UNVOICED-POST-ALVEOLAR-FRICATIVE-CONSONANT phone was deleted
$o:-	long CLOSE-MID-BACK-ROUNDED-VOICED-MONOPHTHONG phone was deleted
$n-+	an APICO-ALVEOLAR-VOICED-NASAL-CONSONANT phone was deleted the current word is a function word
##m	beginning of word, LABIAL-VOICED-NASAL-CONSONANT phone was elicited

Fig. 13. Part of the state transition diagram that interprets Kiel-1 phonetic level annotation labels as well as the full interpretation of the annotating label string "##Q- $a- $l- $s-+ ##S- $o:- $n-+ ##m".

The decision regarding which properties are deviating can be made from a minimal information storage point of view. However, computational aspects and representation preferences can also be contributing factors. By increasing class diversity the modelling of local properties can be reduced, e.g., envision a unique class for an [æ] (IPA) phone which is normally voiced but has been whispered, whose duration is `LONG`, and exists in some specific phonetic context. For the latter to be realised a very large number of classes would have to be defined. This would in turn cause limitations of the computing environment to manifest themselves at some point. On the other hand, decreasing class diversity would cause the system to approach an attribute-based model, such as in a RDBMS, where it is more difficult to associate explicit knowledge with instances. Adding deviating properties preserves diversity without leading to class number explosion.

Finally, fuzzy membership values can also be employed in applications such as speech recognition where, e.g., a phone that is being classified has its class membership expressed with a real value in the set [0,1].

4.3.3 Query-Forwarding

When modelling local properties is combined with the concept of hierarchical scale of structural levels, described earlier, efficient means for indicating properties for objects can be conceived. When an instance is questioned for some property and no *local* and no *general class behaviour* is specified, then the request can be forwarded along another specified dimension, e.g., the hierarchical scale. This permits another object of a different class or level to fulfil the request. This *query-forwarding* technique is used to allow an annotation to have exceptions for some of its properties which are inherited through hierarchical scale. For example, the default language for an annotation may be Finnish but a loan word within the same utterance can be marked as English. When a Finnish word is queried, no local or general class behaviour regarding language is found. Therefore, this same Finnish word forwards the query to its sentence object that, in this example, would have the value `FINNISH` associated with it. However, when the loan word is queried, then its local language property, having the value `ENGLISH`, is returned immediately. Class and "hierarchical scale" inheritance function basically in the same manner; however, an extra degree of freedom beyond the normal class-inheritance paradigm is offered.

4.3.4 Compiler

With the resources and access techniques defined above, a generic compiler suitable for many corpora and target theories can be created. The operation of the compiler, as currently implemented in QSSDB, is best shown by an example where the Kiel speech signal and related annotations described in section 4.2.2 are used. It should be noted that the compiler does not presuppose that corpora always supply certain information, e.g., the acoustic waveform, since some corpora may only have F0 or certain physiological parameters available. The compiler's operations can be divided into two sequential operations: a) generating a suitable representation framework where its output units are stored, and b) interpreting GRF objects and creating new representation units. These two phases are now described.

The compiler first creates an empty representation framework object where its output units can be stored. It then begins to interpret the top-level GRF object that was generated by the corpus parser that, in QSSDB, is a signal since it is a conveniently central object. The signal contains properties expressing its domain (acoustic) and level (waveform). A new domain called *acoustic* is therefore created containing a *waveform* level, which is linked back to the signal, and placed into the new framework. Linked with the signal object is an annotation GRF object containing separate lists of GRF objects for each annotation level, some of which can be seen in Fig. 8. The compiler notices this and begins its analysis. First, existing annotation levels are determined and mapped to possible existing corresponding names in the supported target theory. If some additional levels of annotation are computable from the existing annotations, e.g., syllabification from word level information, and are desired by the user, then they can be generated by suitable resources and included into the evolving framework at this point. In this Kiel-specific case, `PARAGRAPH`, `SENTENCE`, `PHRASE`, `WORD`, and `PHONE` have mappings (found in the CSM) to `ORTHOGRAPHIC-PARAGRAPH`, `ORTHOGRAPHIC-SENTENCE`, `ORTHOGRAPHIC-PHRASE`, `LINGUISTIC-WORD`, and `PHONETIC-PHONE`, respectively. If a mapping is not found then this part of the target theory needs to have an additional resource defined to support it, or else this domain or level will simply not be generated. Next, an empty level is created for every mapping that was found to be supported; if the required destination domain already exists in the framework the level is added to that domain, otherwise, a new domain is created first and then the level added. Specifically, an `ORTHOGRAPHIC` domain is created in which the levels `ORTHOGRAPHIC-PARAGRAPH`, `ORTHOGRAPHIC-SENTENCE` and `ORTHOGRAPHIC-PHRASE` are attached. A `LINGUISTIC` domain containing a `LINGUISTIC-WORD` level, as well as a `PHONETIC` domain with a `PHONETIC-PHONE` level, are also added to the framework. During creation of the individual levels, links back to the GRF object lists are given to the representation levels so they later will know where to find their GRF source objects. Finally, the generation of parallel levels of representation in the other existing domains, i.e., target theories, is performed. In this example the existence of `ORTHOGRAPHIC- PARAGRAPH`, `PHRASE`,

and SENTENCE levels causes similar levels to be created in the LINGUISTIC and PHONETIC domains. The LINGUISTIC domain in turn triggers WORD levels to be created in the ORTHOGRAPHIC and PHONETIC domains. The PHONETIC domain's PHONE level does not cause an ORTHOGRAPHIC-PHONE level to appear since this level is not supported by the orthographic description of the data. However, a new level is created in the LINGUISTIC domain but is called a PHONEME level since this is the PHONE level's hierarchical equivalent. This completes the first stage of compilation where a framework was created containing domains and empty levels for all direct as well as parallel representations.

The second stage of compilation interprets annotation data, builds units, and stores them into the levels of the framework. The compiler runs through each domain and level interpreting the annotation GRF object lists that were linked to levels that have actual data. For example, when the compiler reaches the highlighted GRF object containing the labelling string shown in Fig. 8, the following operations occur. The compiler:

- questions the GRF object for its character set, phonetic alphabet and labelling syntax
- selects the required resources, e.g., ISO-8859-1, German-SAMPA, Kiel-1
- calls the Kiel-1 syntax interpreter which processes the substrings

The observations for the label string are produced, as seen in Fig. 13, and the saved observations are used later on during the linking stage. The compiler builds at this same moment parallel units and adds them to their respective domains, copying temporal information, if possible. Compiling continues until all GRF objects in the domain levels have been processed. It should be noted that no temporal information has been utilised yet in framework generation—although it is available within specific units, e.g., the PHONE unit [m].

4.3.5 Linker

In QSSDB a link can be created between any two or more units. A link is realised as a categorised reference to another object (or a list of objects to represent a one-to-many relation), or can be a typed object itself. Frameworks including units, levels, domains, and links can exist in QSSDB's persistent OO-DBMS and can be utilised by different applications repeatedly—since utilisation is a non-destructive operation—and in separate processing sessions.

Generation of links is performed by the linker which is first given a representation framework containing domains, levels, and units. By using information residing in the CSM—defined by the database designer to accommodate some target theory, or by program code supplied by the user—the linker creates links between units. For example, a rule-based system can be used to build associations

between phones and phonemes. Domains and levels not suggested by the CSM may also be created and linked by user-supplied code, e.g., a rule-based system to generate syllables from orthography.

Once a representation framework has been created in QSSDB, additional explicit representations can be inserted between items in two different ways: i) by adding explicit links between items, or ii) by defining a method function to computationally generate the relation. In this way all items of a speech corpus can be linked to each other explicitly, if required, from the querying application's viewpoint. Since all relationships can be made explicit, the representation frameworks, or parts of them, can be dumped using a different format to a file system, e.g., an OO-DBMS, an RDBMS, an XML representation, etc.

The bottom-right pane of Fig. 9 shows graphically the concept of a representation framework while Fig. 14 shows in detail part of the current example of Fig. 8 from the Kiel Corpus of Read Speech. In Fig. 14 only the `LINGUISTIC` and `PHONETIC` domains of the representation framework generated by the compiling and linking process are shown. Intra- and inter-domain links that indicate some of the annotated correlations occurring between the two domains in this section of the signal can also be seen. Representation units are labelled with a symbol for the reader's visual identification purposes only: `PHONE` and `PHONEME` units are labelled with their German-SAMPA phonetic alphabet symbol, and with a sequence of these symbols for `PHONETIC WORD` units. Other hierarchical units are labelled with orthography for ease of reference.

Links can be created in at least three different dimensions within a representation framework. Two of these cases are when intra-domain links are used to relate units within the same domain:

- Vertical linking of units that are structurally organised within a domain (target theory) according to a predefined hierarchical scale. Hierarchical relationships are identified by linking objects in the same domain but on adjacent levels. For example, the [m] phone unit seen in Fig. 14 is therefore linked to its next higher level of scale, e.g., the phonetic word unit [m I]. In TIMIT and Finnbase, where hierarchically lower phonetic segment levels exist, closures and releases of stops are linked to their respective phones.

- Horizontal linking of units that are organised in a level according to a specified order or temporal information. These relationships are identified by linking neighbouring units in the same level. For example, the [m] phone unit is linked to its previous phone unit [aU] and to its next phone unit [I].

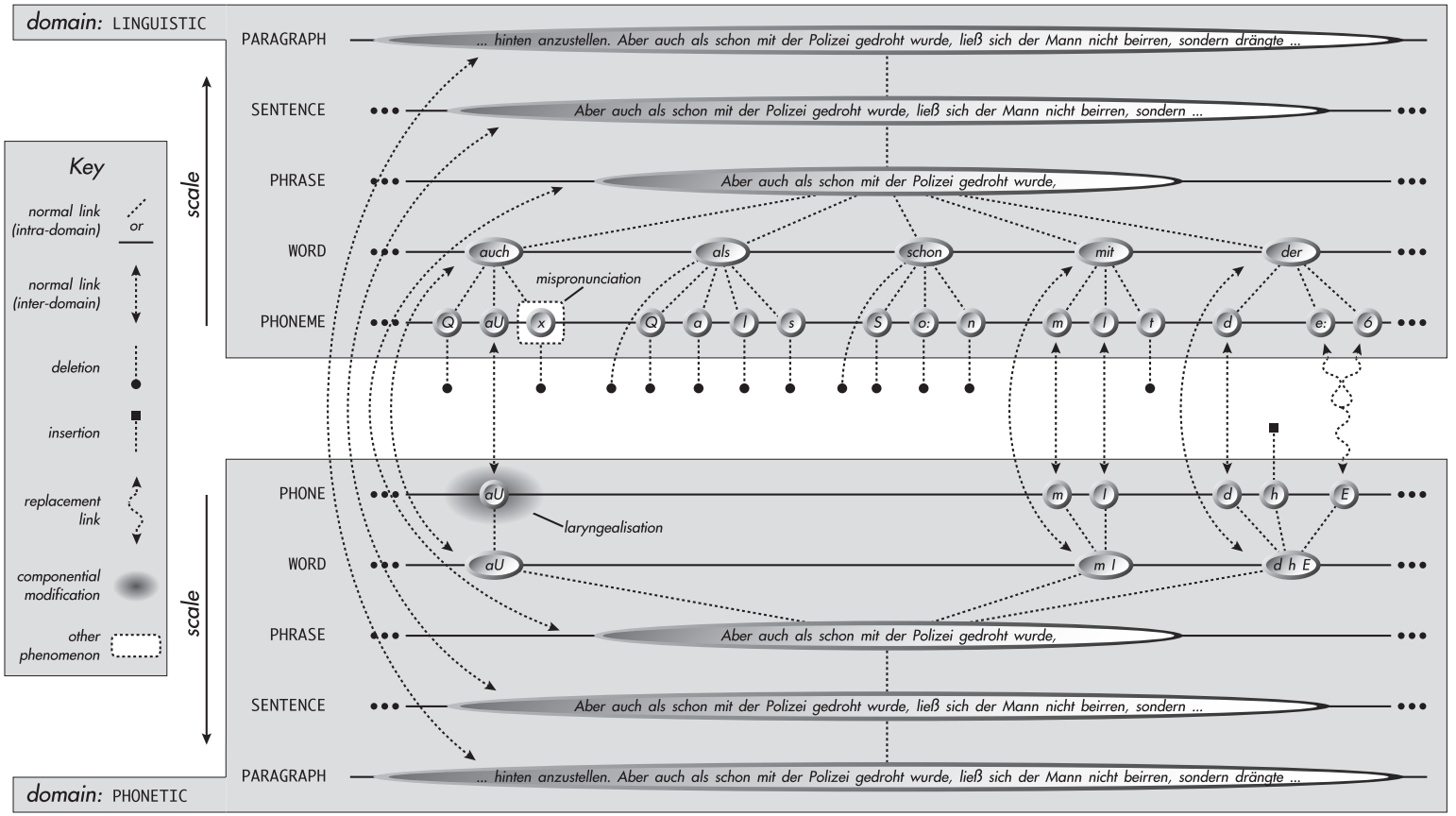


Figure 14. The LINGUISTIC and PHONETIC domains of a representation framework. Inter-domain links relate order and hierarchy within a domain while intra-domain links indicate some of the annotated correlations occurring between domains.

On the other hand, inter-domain links associate units that have mappings between domains.

- Representation units in different domains are linked, if possible, to indicate their realisation in another domain. For example, linguistic phoneme level and phonetic phone level units can be related to each other to indicate whether a phone was elicited as expected or whether an insertion, deletion, or replacement occurred.

4.3.6 Accessing Links

Representation units can access other units by:

i. vertical links via the hierarchical tree structure,
ii. horizontal links that access units according to some order on the same level, or
iii. inter-domain links.

4.3.6.1 Vertical Links

Vertical links are utilised in the following manner. The generic function *superstructure* can be applied to any representation unit in any level or domain. The unit responds by returning the unit (or units) it is linked to at a higher level. This generic function can also be requested to return the superstructure at some predefined level, e.g., when applied to the left-most `PHONEME` unit in Fig. 14 labelled with *Q* and denoted by `x`, the form:

```
(superstructure x :level 'sentence)
```

returns the *Aber auch als schon ...* `SENTENCE` unit. Likewise, when the generic function *substructure* is applied to a unit it returns as a list all of its units at the next lower hierarchical level, or at some specified lower level.

4.3.6.2 Horizontal Links

Horizontal links are accessed via the *next-unit* and *prev-unit* generic functions. When applied to a unit these functions return their next or previous neighbouring unit according to level order. These functions can be specified to return a unit at a specified level as well, e.g., when the form:

```
(next-unit x :level 'word)
```

is evaluated, where x is the [m] PHONE unit in Fig. 14, the PHONETIC-WORD unit [d h E] is returned. Explicit links allow an object to be identified immediately but memory is required to represent the link. When temporal annotations are processed in QSSDB, horizontal links are typically implemented by computational links. Computational links, i.e., virtual links that are implicit in the representation, are slower in terms of access time but require less memory since no link structure is involved. The use of computational links is motivated by simplified processing when the framework structure is modified, e.g., when a unit is inserted into a level.

4.3.6.3 Inter-Domain Links

In Fig. 14 four unique types of inter-domain relations are found:

i) Normal relations, where the canonical form was elicited by the speaker as expected. These are seen as lines or arcs with arrows on both ends. For example, the linguistic word unit with the symbol *auch* is related to its phonetic word counterpart [*aU*] through a normal link. The *auch* word unit, represented by the symbol x, can access its word counterpart in the phonetic domain via the form:

```
(unit x :domain 'phonetic)
```

ii) Deletions, where the canonical form was not elicited. Linguistic units exhibiting deletions in their phonetic realisation are shown with links ending in a circle. For example, the left-most phoneme unit in the linguistic domain labelled *Q* is not related to a phonetic unit. When queried with the same above form the value NIL is returned indicating that no phonetic realisation exists. Also noteworthy are two consecutive linguistic words which have a depleted phonetic domain representation since a sequence of seven phonemes was not realised. Similarly, if either of these linguistic words are queried for their phonetic word counterpart then NIL is returned.

iii) Insertions, where a phonetic unit was elicited that was not expected in the linguistic form. For example, an unvoiced glottal fricative consonant (labelled with the symbol *h*) was inserted by the speaker when speaking the linguistic word unit /*d e r*/, as shown in the example with a line ending in a square. Symmetric to the deletion case, insertions can be detected when the following form returns NIL:

```
(unit x :domain 'linguistic)
```

iv) Replacements, where a unit is replaced by an unexpected unit in another domain. One such example is seen in Fig. 14 at the right-hand side where the linguistic form contains two monophthong phonemes /e:/ and /6/. This phonemic sequence was expected to be realised as one diphthong [e:6], a CLOSE-MID-FRONT-UNROUNDED → NEAR-OPEN-CENTRAL-UNROUNDED voiced transition. However, the speaker uttered [E] instead, an OPEN-MID-FRONT-UNROUNDED-VOICED-MONOPHTHONG. When the above Lisp form is applied to the [E] phone unit, a list structure containing both related phoneme units is returned.

4.3.7 Accessing Local Properties

In general, any local property required to model the detail present in an annotation may be associated and stored with a unit. For example, in Fig. 14 a componential modification exists for the PHONE unit [aU] since a remnant of laryngealisation still exists. A local property such as glottalisation for a unit x can be queried by the form:

```
(laryngealised-p x)
```

Other componential modifications, such as nasalisation, velarisation, and palatalisation can be detected similarly with appropriate predicate functions that return either T or NIL. Finally, other phenomena, such as the PHONEME unit marked with a mispronunciation in Fig. 14, can also be modelled by having the compiler tag the unit in question.

4.3.8 Other Units and Parallel Representations

A *di-unit* can be specified between any two adjacent units on the same level, e.g., a DIPHONE relates two consecutive PHONE units. A diphone's temporal extent is defined to begin from the temporal mid-point of the first phone and end at the temporal mid-point of the second phone. In QSSDB, a diphone object carries no temporal information but rather is able to calculate its extent from the phones it is linked to. For applications requiring high efficiency, the temporal extent can be cached with the diphone object itself at the cost of increased memory utilisation.

Figure 15 shows an example where both diphone and diword levels have been generated. These parallel representations are used to define boundary-centred perspectives of data as compared to typical segment-centred views. *Tri-units* can be defined in a similar manner to model unit triples. The need for

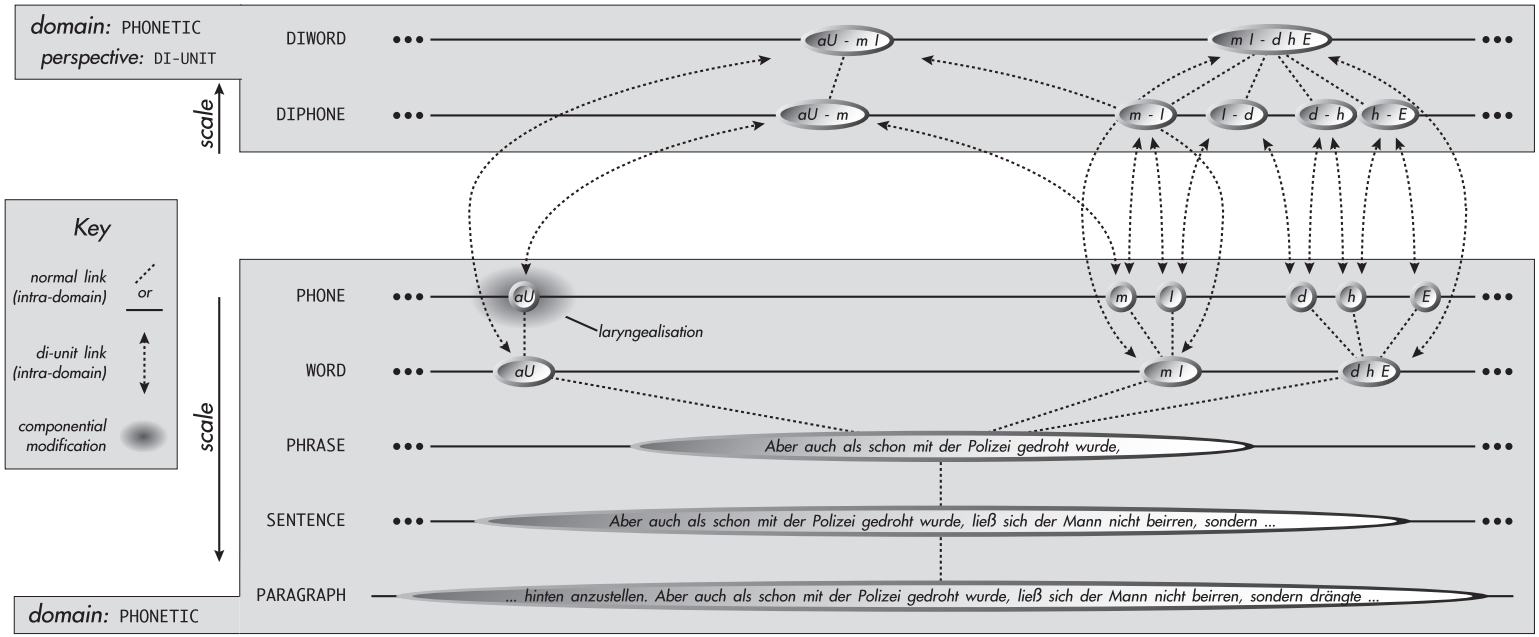


Figure 15. The PHONETIC domain of Fig. 14 along with a parallel DI-UNIT representation. DIPHONE and DIWORD units are generated to yield boundary-centred perspectives of the data at specific levels of hierarchical scale.

a certain perspective depends largely upon application requirements during the database access phase, e.g., diphones have been found to be convenient working units when studying diphone recognition (Altosaar & Karjalainen, 1992).

4.3.9 Framework Visualisation

Representation frameworks for corpus annotations can be envisioned in a three-dimensional space. Visualisation is useful for verifying the operation of the compiler, the linker, and in designing database queries since complex environments can be drawn explicitly.

4.3.9.1 Visualisation of Temporal Frameworks

The representation framework for the sentence "Bald ist der Hunger gestillt.", from the Kiel Corpus of Read Speech is shown in Fig. 16. Four domains have been instantiated in this framework: starting from the bottom and proceeding in a clockwise direction:

i. `ACOUSTIC` (includes the `WAVEFORM` level),
ii. `PHONETIC` (includes `SENTENCE`, `PHRASE`, `WORD`, and `PHONE` levels),
iii. `LINGUISTIC` (includes `SENTENCE`, `PHRASE`, `WORD`, and `PHONEME` levels), and
iv. `ORTHOGRAPHIC` (includes `SENTENCE`, `PHRASE`, and `WORD` levels).

As can be seen, each of the three symbolic domains, which excludes the numeric domain `ACOUSTIC`, has levels which contain representation units.

Visualisation is carried out by building a 3-dimensional representation of the representation framework. An integrated, Lisp-based interface to the QuickDraw-3D (Apple, 2001) rendering system is included in QSSDB (Altosaar, Karjalainen & Vainio, 2001).

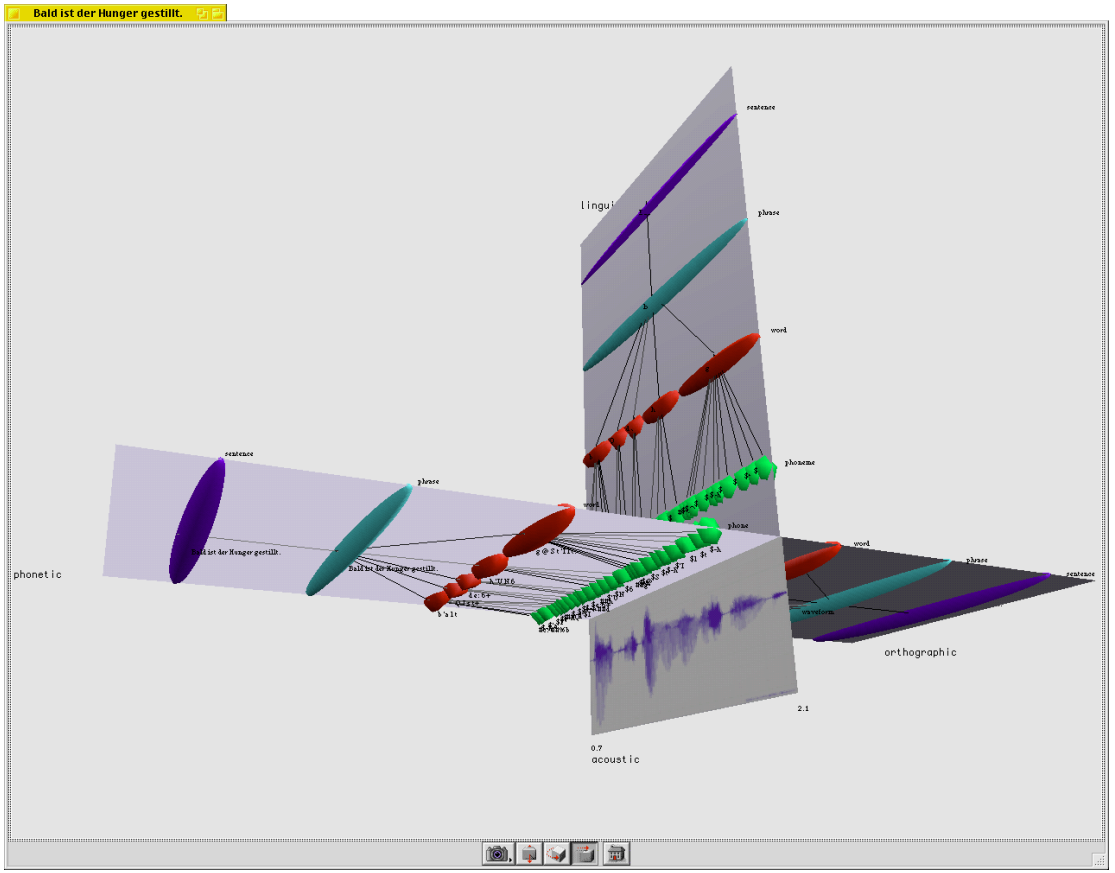


Fig. 16. Visualisation of the representation framework for the Kiel sentence "Bald ist der Hunger gestillt." From the bottom and working clockwise: the ACOUSTIC, PHONETIC, LINGUISTIC, and ORTHOGRAPHIC domains which are composed of separate levels consisting of speech units. Database queries operate on the units as well as their links. Inter-domain links exist between some units as well, e.g., between a phone and its related phoneme, and can be used to detect deletions, replacements, and insertions.

4.3.9.2 Visualisation of Atemporal Frameworks

Atemporal annotations—collected during the recording process of a corpus—deal with data that are not a function of time. They may include a history of each speaker, a detailed description of the recording equipment, the physical recording setup, recording anomalies, etc. Traversal through atemporal frameworks is accomplished by links as is the case with temporal frameworks. In atemporal annotations, the relations between different units do not necessarily follow a natural hierarchy, e.g., as in phonetics. Therefore, user desired explicit relationships between unit classes are specified manually by an association table. In QSSDB, the system automatically generates the required explicit links and computationally shortest-path methods from the association table. This gives every unit in the framework knowledge on how to navigate most effectively to any other unit.

For visualisation purposes each unit class is assigned a representative token, e.g., talkers, microphones, books (representing speaking tasks and talker histories), reflective surfaces, etc. The visualised tokens are linked to the representation units allowing for user manipulation and interaction. Figure 17 shows the framework depicting the atemporal annotations of the `ANDOSL MMK_FMK1` corpus where 17 speakers participate in MAP tasks (semantically-constrained spontaneous speech). Speaker `S217`, seen at the bottom of the stack of talkers, was requested to indicate where information related to him existed in the rest of the corpus: blue lines indicate one-to-one links, green lines one-to-many links, and red lines one-to-many links in both directions. Such requests can be used effectively, e.g., whenever a sub-corpus containing only one speaker needs to be created since it must be known what other peripheral objects need be saved to retain sub-corpus integrity.

A set of CD-ROMs that compose a large corpus can be merged to share unique objects, e.g., a microphone that is used in all of a large corpus' recordings is only modelled once. Since units know of their relations to other units a microphone, e.g., can determine all the speakers who have ever spoken into it. Finally, atemporal and temporal frameworks can be linked with one another allowing for corpora wide database queries. For example, in the bottom centre of Fig. 17 the stack of dark-green `RECORDING` objects represent temporal representation frameworks.

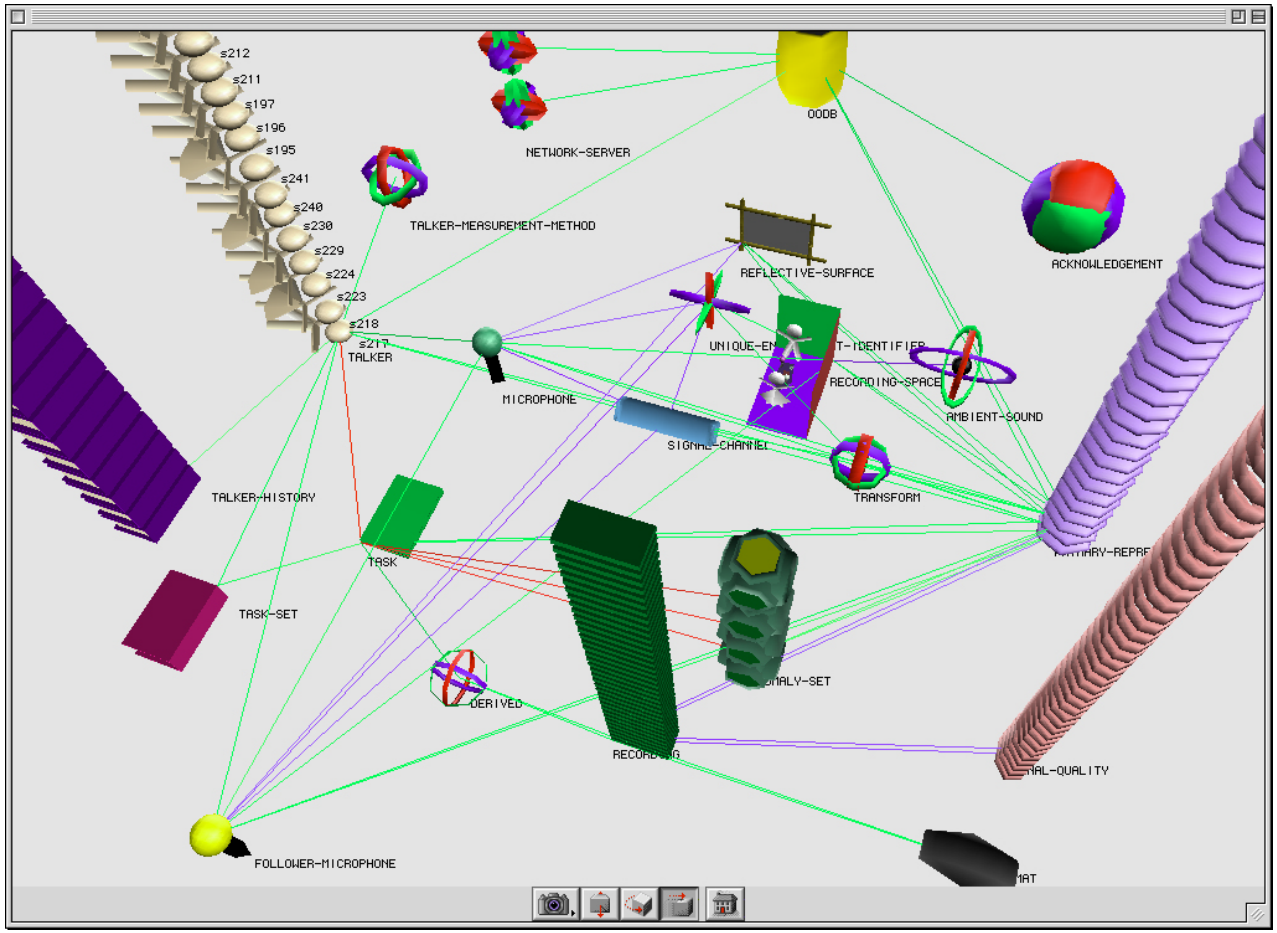


Fig. 17. Visualisation of ANDOSL MMK_FMK1 speech corpus' atemporal annotations. Representation units are stacked vertically. Talker S217 indicates his sphere of influence by explicit and computational links.

5 Uniform Database Access

Concurrent identification of similar parts of signals and annotations, from one or more—possibly different—corpora, is desirable since inferences can then be made, e.g., between the characteristics of a signal (or its derived features) and its symbolic annotations in a diverse and possibly multi-lingual environment. The goal of this thesis is to offer a database access formalism that is uniform and independent of specific corpora—an environment where database queries can be performed in an intuitive and efficient manner. This section describes ways in which the representation framework can be queried.

5.1 Search Mechanism and Tests

Since a representation framework models the characteristics and relationships of speech units explicitly, it offers a good foundation on which to perform database queries. Contextual patterns that include specific unit characteristics can be identified within a framework by a structure matching search function. This is accomplished by traversing a representation framework through existing links and successively applying a search function to each unit that determines whether the local context matches the desired one. The function *find–units* provides this mechanism and operates on two arguments: the unit to perform the search on, denoted by `x`, and a test:

```
(find-units x test)
```

Since `find-units` is a generic function its first argument may be a representation unit, level, domain, framework, signal, annotation, database, or a list containing any of these items. `find-units` returns the units that satisfy `test`, i.e., the result of a query is a list of actual units within the frameworks, e.g., a list of active pointers into the structures. This permits the units to be used flexibly in further processing operations since all data and relationships are available for use via links.

The second argument required by `find-units` is a test. Expressed as a function, `test` is given to each unit that then executes the function within its own contextual environment. The test function may reference any of the unit's class-based characteristics, local properties, and links to other units to determine the closeness of a match. Test functions return `T` if a match occurred and `NIL` otherwise. They may also return a real valued scalar or a vector of scalars indicating how well the test matched a local environment. Continuous valued matches can then be sorted according to some criterion before further analysis takes place.

5.2 Examples of Uniform Database Access

In this section illustrative examples of queries are given that can be used to find desired contexts in representation frameworks. Queries are formulated by the user writing Lisp functions that operate on the items found within a representation framework.

5.2.1 Query involving a Sequence of Phones or Phonemes

A desired pattern in a sequence of phones, e.g., a consonant, followed by a close, back, rounded, voiced monophthong phone [u] (Worldbet), followed by an unvoiced, apico-alveolar, fricative [s] (Worldbet), can be viewed graphically as seen in Fig. 18.

Fig. 18. Graphical depiction of a [consonant]–[u]–[s] sequence of phones.

Functionally, the above pattern for the specified sequence of phones may be expressed as a predicate function in Lisp as:

```
(defun C/u/s-p (x)
  (and (typep x (ARPABET "uw" :phone))
       (typep (prev-unit x) 'consonant)
       (typep (next-unit x) (MRPA "s" :phone))))
```

where:

- the Lisp macro `defun` defines a new function named `C/u/s-p`
- the argument to the function is `x` which can be any type of speech unit
- at function compile-time the macro `ARPABET` evaluates its two arguments (the string `"uw"` and the class specifier `:phone`) and produces the symbol `CLOSE-BACK-ROUNDED-VOICED-MONOPHTHONG-PHONE`; similarly, `MRPA` (Machine-Readable-Phonetic-Alphabet, see Table 1) evaluates its arguments to `UNVOICED-APICO-ALVEOLAR-FRICATIVE-CONSONANT-PHONE` [2]
- the function `(typep x y)` tests if instance `x` inherits class `y`; returns `T` or `NIL`
- the functions `prev-unit` and `next-unit` access neighbouring units
- the Lisp macro `and` performs the logical 'and' function

[2] This compile-time technique can be seen as a simple form of *metamodel-driven interaction*—where an application indirectly accesses data by first accessing the metadata (in this case the Worldbet hierarchy) and then formulates the query to access the data (Blaha & Premerlani, 1998).

When function `C/u/s-p` is applied to a representation unit, a check is first made to see that `x` inherits from the class `CLOSE-BACK-ROUNDED-VOICED-MONOPHTHONG-PHONE`. If so, `typep` returns `T`, otherwise `NIL` is returned and the logical `and` function causes `C/u/s-p` to return `NIL` immediately—indicating a failed match. Otherwise, the previous unit on the same level as `x` is tested to see if it inherits the class `CONSONANT`. If it does not, `C/u/s-p` returns `NIL` immediately, indicating a non-matching context. Otherwise, the function continues to see if the next phone is an [s] (MRPA): if so, the function returns `T`, otherwise `NIL`. Note that the mixing of phonetic alphabets in queries is permitted since class-based distinctive features are referred to and not strings.

The function `C/u/s-p` is applied to all five corpora referred to in Table 1—whose utterances are first transformed into representation frameworks—by supplying the test as the second argument to the `find-units` function. (In this example `units` tested are either `PHONEME` or `PHONE` units, depending upon the corpus; ANDOSL and Estonian use phoneme annotations while Finnbase, Kiel, and TIMIT use phones. Otherwise, the query is identical for this set of corpora: `:phoneme` is substituted for `:phone` in the definition of `C/u/s-p` for the ANDOSL and Estonian cases). The results are presented in Table 2 in tabular form.

Table 2. Results of applying the `C/u/s-p` test function to five different corpora where frameworks indicate the number of utterances existing in each collection of speech. Matches indicate the number of contexts found that satisfied the test function's criteria.

Corpus	*Frameworks*	*Units Tested*	*Matches*
ANDOSL (YB)	*2 400*	*112 918*	*89*
Estonian (MTS1)	*130*	*13 312*	*114*
Finnbase	*4 703*	*62 378*	*101*
Kiel (Read)	*3 876*	*149 487*	*26*
TIMIT	*6 300*	*230 639*	*18*

The units that the search function identified can be processed further immediately without having to save intermediate results to a file system. For example, Fig. 19 shows the temporal duration distribution and average auditory spectrum of the matching units for each different corpus. These were calculated in the same integrated Lisp-based QuickSig environment.

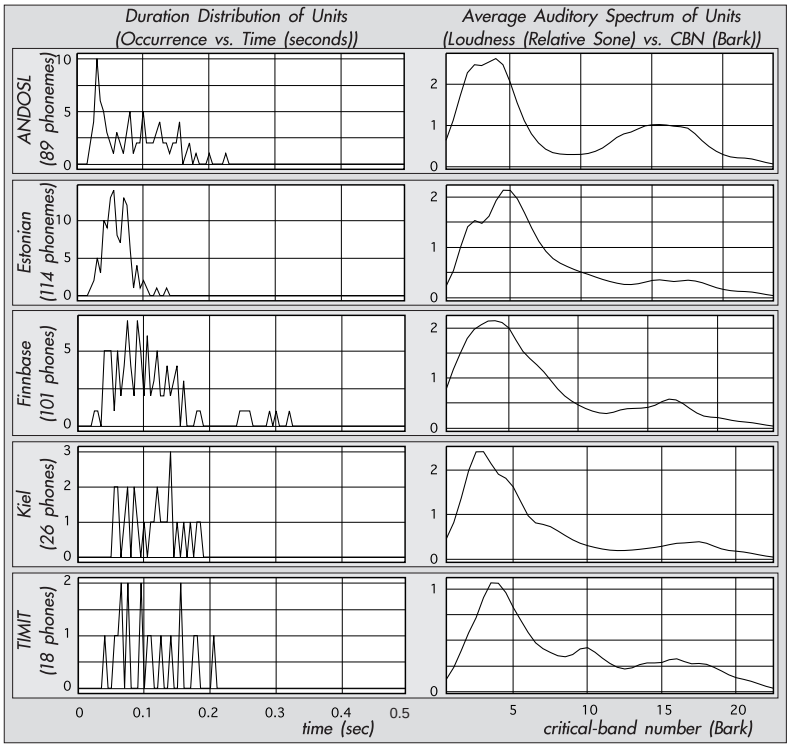


Fig. 19. Further processing in the same QuickSig environment of phone and phoneme contexts that matched the `C/u/s-p` query function: a distribution of temporal duration and an average auditory spectrum is shown for each set of matching units.

5.2.2 Query involving Morphology

In this example it is desired to find all spoken word units in a corpus that are a copula (linking verb) and are followed immediately by an adjective, as can be seen graphically in Fig. 20.

Fig. 20. Graphical depiction of a linking verb followed by an adjective.

Formulating this query in Lisp using the `find-units` function is as follows. `find-units` requires two arguments: a root unit from where to begin testing and a test. In this case `*abstract-node*` is a user defined node that is updated by the user in a QSSDB database grapher display. Its value can be used to specify which set of data to apply the query to. The second argument to `find-units`, the test, need not be a named function but can be a simple lambda expression (an unnamed function) instead:

```
(find-units *abstract-node*
   #'(lambda (x)
       (and (typep x 'phonetic-word-unit)
            (copula-p x)
            (adjective-p (next-unit x)))))
```

In this example `x` represents any unit `find-units` applies `test` to. For the test to return a non-`NIL` value three conditions must be met: i) the current object must inherit from the `PHONETIC-WORD-UNIT` class, ii) `x` is a copula, and iii) the next unit with respect to `x` is an adjective. When applied to a 692 utterance Finnish corpora that has part-of-speech tags, `find-units` returns a list of 63 word units that can be used for further processing. For example, applying the function `play-unit` to the returned list of words allows the user to listen to the matches audibly.

5.2.3 Query involving Recursion

In this example we desire to find the average unit rate of a phone in its actual context, i.e., to determine the number of phones in a window centred around the phone in question. As a default, the window size is set to one second. Therefore, starting at some phone we need to traverse backwards in time finding how many phones (or portions thereof) are within a 500 ms window. A similar process is required to process the phones forward in time. This type of query is data dependent since the number of phones traversed will vary and is well solved using a recursive query, as shown in the following Lisp definition. As can be seen, the generic function `unit-rate` defines two mutually recursive functions, `find-left-edge` and `find–right–edge` using the `labels` construct. These functions perform the actual traversal from unit to unit until a window edge has been reached, updating local variables specified in the `let*` special form, as needed. Finally, when the edges and number of units have been found, the average unit rate is calculated.

When applied to the 692 plainly spoken sentences of a 1000 utterance Finnish corpus, 44093 phone units had their average phone rate calculated over a one second window. Represented as a histogram, the distribution of average phone rate can be seen in Fig. 21.

Even though `unit-rate` utilised two recursive functions, the functions `prev-unit` and `next-unit` are defined recursively as well. The number of calls to these functions were measured during the repeated queries and were 458 427 for `prev-unit` and 457 978 for `next-unit`. Total execution time on a 300 MHz PowerPC G3 based laptop was 53 seconds which translates to over 17 000 framework queries per second. Without a precompiled and linked framework structure, database access time would be considerably slower. Even worse performance would be obtained if the querying applications existed as different applications running on the same (or networked) machine and I/O would be through the file system.

```
(defmethod unit-rate ((unit unit-mixin) &key (window 1.0))
  ;;; determine the number of units in window centred around unit
  (let* ((int-length (interval-length (interval unit))))
    (when (< window int-length) (setq window int-length)))
  (let* ((number-of-units-to-left 0.0)
         (number-of-units-to-right 0.0)
         (centre (mid-point (interval unit)))
         (left-edge (- centre (* 0.5 window)))
         (right-edge (+ centre (* 0.5 window))))
    (labels ((find-left-edge (x)
               (let* ((int (interval x))
                      (bp (beg-point int)))
                 (cond ((< left-edge bp)
                        (let* ((prev (prev-unit x)))
                          (cond (prev
                                 (incf number-of-units-to-left)
                                 (find-left-edge prev))
                                (t (setq left-edge bp)))))
                       (t (let* ((over (/ (- left-edge bp)
                                          (interval-length int))))
                            (decf number-of-units-to-left over))))))
             (find-right-edge (x)
               (let* ((int (interval x))
                      (sp (stop-point int)))
                 (cond ((> right-edge sp)
                        (let* ((next (next-unit x)))
                          (cond (next
                                 (incf number-of-units-to-right)
                                 (find-right-edge next))
                                (t (setq right-edge sp)))))
                       (t (let* ((over (/ (- sp right-edge)
                                          (interval-length int))))
                            (decf number-of-units-to-right over)))))))
      (find-left-edge unit)
      (find-right-edge unit)
      (/ (+ number-of-units-to-left 1 number-of-units-to-right)
         (- right-edge left-edge)))))
```

As seen above, the generic function `unit-rate` is defined for any instance that inherits the class `unit-mixin`. Since all representation units in QSSDB inherit this class, the same method can be applied to any other type of unit as well, e.g., segment, phone, phoneme, syllable, word, sentence, etc., existing in any domain. For example, the average word rate could instead be calculated over some desired averaging window by applying `unit-rate` to a set of words instead of a set of phones, as was done previously. Recursive search functions operating on representation frameworks allow the user to define complex functions in a natural object point-of-view manner, many of which cannot be formulated, or processed efficiently, in a conventional relational-database model of speech. Finally, libraries of search functions can be constructed readily since existing functions can be used in new search functions.

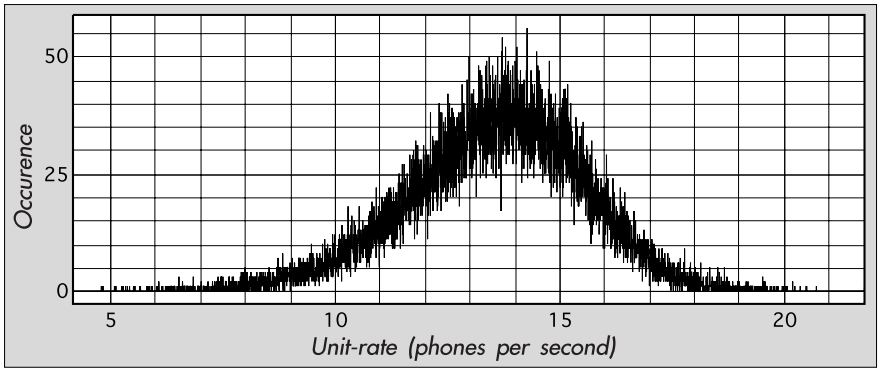


Fig. 21. Distribution of average phone rate (units vs. phones per second) calculated over 44093 phones.

5.3 Modelling a Specific Corpus: ANDOSL

In this section the ANDOSL corpus of speech is modelled using two different database representation techniques: RDBMS and OO-DBMS. Identical queries are then applied to the same speech material in an entirely parallel way to reveal differences in query complexity and relative performance.

5.3.1 The ANDOSL Corpus

The ANDOSL data corpus has several components including isolated words, phonemically-rich sentences, and spontaneous speech. These data were collected from a total of 264 speakers in either a single-speaker reading situation or dual speaker dialogue situation (Millar *et al*., 1994, Vonwiller *et al*. 1996, Millar *et al*., 1997). Each speaker contributed on the order of 80 isolated words including spoken digits, common consonantal context vowel exemplars, and some other key words. Native speakers of Australian English contributed 200 sentences designed to include all phonemes and all second-order pairings of phonemes of the language. Migrant speakers, who comprise some 22% of the Australian population, contributed 50 sentences that were derived from the 200 but whose second-order pairings were restricted to at least one exemplar of each pair of broad phonemic classes. Furthermore, each speaker was paired with another speaker for two recordings of structured but free flowing dialogue. For the queries in the following sections, data was restricted to one of the 30 CDs which make up the ANDOSL corpora. The sub-corpus in question, labelled "YB_Sentences", includes 200 sentences each from six male and six female speakers for a total of 2400 utterances.

5.3.2 RDBMS Modelling

Access to ANDOSL via database systems using relational structures has been explored in at least two ways:

i) At Macquarie University the linguistic structure of the data has been modelled using the EMU system (Harrington *et al*., 1993) which comprises a layered relational structure ranging from an acoustic-phonetic level, up via a phonemic level to a syllabic level, and eventually reaching an orthographic level. This structure may be traversed in either direction.

ii) At the Australian National University (ANU) the description of the source and quality of the data files comprising the speech signals has been modelled using the regular tables of an RDBMS system. It is this form whose raw data is distributed on the ANDOSL CD-ROMs.

The RDBMS system at ANU is based around a Notional Spoken Language Description (NSLD) scheme first described in (Millar, 1992). NSLD has developed towards a comprehensive scheme (Millar, 1998) and thus the ANDOSL RDBMS is a subset. It has been used extensively for the selection of data for issuing on CD-ROMs and has been made available through a limited set of WWW forms (http://andosl.anu.edu.au/andosl) to users of the data who are searching for something specific within it, or who wish to request further ANDOSL data that was not included in the CD-ROM releases. Some additional tables have been added to service these particular needs such as a `QUALITY` and an `ANNOTATION` table which provide information about the signal quality of each signal file and the presence of phonemic annotation data. Table 3 lists the RDBMS tables.

It should be noted that the relational scheme necessitated two independent tables for `SPEAKER` and `SPEAKER_HISTORY` where the former contained all data that described their status at the time of recording as illustrated in Table 4, whereas the latter contained all data referring to relevant episodes within their life as illustrated in Table 5. Each episode was modelled using five variables but the number of such episodes was highly variable. Using both tables linked by the speaker identifier (SID) column in each, current speaker characteristics and the speaker's prior speaking experience could, for instance, be used for selection.

Table 3. RDBMS tables currently implemented at ANU for the description of ANDOSL.

```
DDF(SOURCE)         Links Data, Speaker, Material ids; date,time,length
FORMAT              Extensive information on file format
LOCATION            Extensive information on file location
TRANSFORM           Information on post-digitisation transforms
DERIVED             Information of additional related data
ANNOTATION          Info. about the method of phonemic annotation
ENVIRONMENT         Information on the recording environment
ANOMALY_CODES       Info. on anomalies occurring in assembling data
QUALITY             Info. on signal level,distortion,background noise
SPEAKER             Extensive static information about the speaker
SPEAKER_HISTORY     Episodic info. about the speakers life history
```

Table 4. NSLD descriptors for static speaker characteristics.

```
SID:  <speaker_identifier>
%------------------------------------ MEASUREMENT METHODS
MET: <filename.met> % methods for all anatomic measures
%------------------------------------ PERSONAL/PHYSICAL
SEX: <sex_of_speaker>                  % "m" or "f" only
YOB: <year_of_birth>                   % 'four digit number'
LEV: <education_level>                 % "primary", "tertiary", etc
HGT: <height>                          % 'four digit number' - unit = mm
WGT: <weight>                          % 'three digit number' - unit= kg
LVT: <length_of_vocal_tract>           % 'three digit number' - unit= mm
COH: <circumference_of_head>           % 'three digit number' - unit= mm
ECD: <eye_to_chin_profile_distance>    %'three digit number'unit=mm
LCP: <lung_capacity>                   % unit = litres
LFR: <lung_flow_rate>                  % unit = litres/minute
%------------------------------------ SPEECH/LANGUAGE
NAL: <native_language>                 % "French", " Spanish", etc
SOC: <sociolect_of_target>             % "general","broad","cultivated"
ACC: <strength_of_accent>              % (wrt) filename or "none"
INT: <intelligibility_level>           % "high"or'filename'.int=ASCII
FLU: <fluency_of_speech>               % fluent, moderate, disfluent
HER: <hearing>                         % "good", "moderate", "poor" only
SPQ: <speech_quality>                  % "normal", "lisp", etc
%------------------------------------ FAMILY INFORMATION
MNL: <mothers_native_language>         % "German", "Russian", etc
MOC: <mothers_occupation>              % "Baker", "Engineer", etc
MPO: <mothers_place_of_origin>         % 'city/country'
FNL: <fathers_native_language>         % "German", "Russian", etc
FOC: <fathers_occupation>              % "Baker", "Engineer", etc
FPO: <fathers_place_of_origin>         % 'city/country'
```

Table 5. SPEAKER_HISTORY descriptors in RDBMS.

```
SID:  <speaker_identifier>
%------------------------------------ SPEAKER HISTORY RECORD
Cnn: <category>                  % 11 categories see below
Ynn: <type>                      % sub-category appropriate to category,
Lnn: <level>                     % hours per week or level '1..3'
Bnn: <start_year>                % first year of episode
Enn: <end_year>                  % last year of episode
%------------------------------------ ELEVEN CATEGORIES
% language, singing, voice training, vocal stress, education,
% residence, music instrument, smoking, medical, sport, employment
%------------------------------------ TYPE EXAMPLES
% language: greek, italian etc.
% singing: opera, rock, professional, amateur, etc
% voice training: purpose
% vocal stress:  teaching, noise in workplace, etc.
% education: primary, tertiary etc.
% residence:  country, city
% music instrument: flute, clarinet etc.
% smoking: cigarettes, pipe
% medical: voice related condition, operation
% sport: netball, cricket, tennis etc.
% employment: teaching, student etc.
%------------------------------------ LEVEL EXAMPLES
% language: 1 = not fluent, 2 = moderately fluent, 3 = fluent
% education:  year three, etc.
% residence:  n/a
% smoking:  number of cigarettes per week or hours per week
% medical:  severity of condition
% other categories:  hours per week
% ------------------------------------ BEGIN / END YEAR
% if Bnn: > 1000  and Enn: > 1000 then value = four digit year
% if Bnn: = 0 or blank and Enn: < 1000 then Enn: = number of years
```

5.3.3 RDBMS Queries

Queries in the RDBMS system using standard query language (SQL) can be written to access one or more of the relational tables. Examples of queries using a single table as well as using four tables to extract data files with specific characteristics relating to their speaker's characteristics, their speakers' history, the material spoken and the quality of the signal, are given.

```
SQL> run
  1  SELECT sid
  2  FROM speaker_history
  3  WHERE cnn like 'smok%' AND ynn like '%igarett%'
  4* GROUP BY sid HAVING SUM(to_number(enn) - to_number(bnn)) > 5

RESULT> SID: s111, s126, s128
```

Query 1 in RDBMS Format: *List the speakers who have smoked cigarettes for a cumulative period of more than 5 years.*

```
SQL> run
  1  select distinct speaker.sid,
  2  from speaker,speaker_history,ddf,quality
  3  where speaker.sid=speaker_history.sid
  4    AND speaker.sid = ddf.sid
  5    AND ddf.dfn = quality.quanum
  6  AND ddf.pid = 's017'                    /* sentence 17 */
  7  AND quality.npl='0' AND quality.nnl='0' /* no clipping */
  8  AND to_number(quality.med)>1800         /* high energy */
  9  AND to_number(speaker.yob) > 1970       /* born after 1970 */
 10  AND speaker_history.cnn like 'mus%'     /* musical_instr. */
 11* AND speaker_history.ynn like 'piano%'   /* sub-cat: piano */

RESULT> SID: s115
```

Query 2 in RDBMS Format: *List all speakers who generate a high energy version of sentence 17 but without any signal clipping and who were born after 1970 and who play the piano.*

5.3.4 OO-DBMS Modelling

The structured descriptive files existing in ANDOSL contain a diverse variety of real-world attributes concerning speakers, recording conditions, file locations, etc. However, details regarding these real-world objects may be scattered among several files creating inherent difficulties in conceptualising the scope of the data, discriminating between variant and invariant information, and determining object relationships. For example, an unnatural dichotomy occurs between `SPEAKER` and `SPEAKER_HISTORY` since episodic information is a complex data type having anywhere from two to five values and is not suitable to be represented in the same RDBMS table with other one dimensional speaker data. This is an example where the representational limitations of the attribute-based RDBMS scheme force data to be

partitioned in a contrived manner.

For these reasons an entity-based object model of ANDOSL was created that would correspond to the real world, simplify the expression of queries, and facilitate rapid database response. The following aspects were observed during its genesis:

i. *Abstraction.* A real-world model was first drawn out without regard to detail that emphasised the natural relationships between objects. For example, since recordings took place within an anechoic-chamber, a *recording-space* object was defined as well as other objects such as *microphone*, *talker*, *speaking task*, and *reflective-surface*—usually the piece of paper on which the task was inscribed. The signal path from the *talker* through *microphone*, signal *channel*, signal *transform*, *primary-representation* file, and finally to *recording*, was made via explicit relationships.

ii. *Encapsulation.* Since external specification was separated from internal implementation, objects were given functional interface methods that facilitated the writing of queries. For example, a database object `x` could be asked "what talkers do you include" by a call to the generic function *talkers*, as in `(talkers x)`, and would return its `TALKER` objects in a list. Likewise, a microphone `x` could be asked to supply all of the talkers who have ever spoken into it by calling `(talkers x)`, or all of its recordings by the form `(recordings x)`.

iii. *Modularity*. Organising ANDOSL data into groups of closely related objects promoted coherence and understandability. Scaling up the design to handle the dialogue tasks with dual-channel recordings did not require a redesign but rather only the addition of some new relations. By shifting from an attribute-based model with 154 parameters (the RDBMS implementation) to an entity-based model (the OO-DBMS implementation), the system became more tractable since only 21 different object classes existed.

One possible object-oriented model that was implemented for ANDOSL data can be seen in Fig. 22. Data was read in from the CD-ROM files and distributed according to this set of object classes. A value-based identity (Blaha & Premerlani, 1998) was chosen to ensure data integrity, i.e., only uniquely valued objects were entered into the network. The system determined automatically, given a set of desired relationships that were chosen a priori, the minimum explicit relation types (indicated by arrows) required between objects, i.e., either a 1-to-1 or 1-to-many relation. Logical real-world modelling was focused on, e.g., in theory a `PRIMARY-REPRESENTATION` (the original audio recording) may have spawned several `RECORDING`s even though this was not the case with these data. This object-

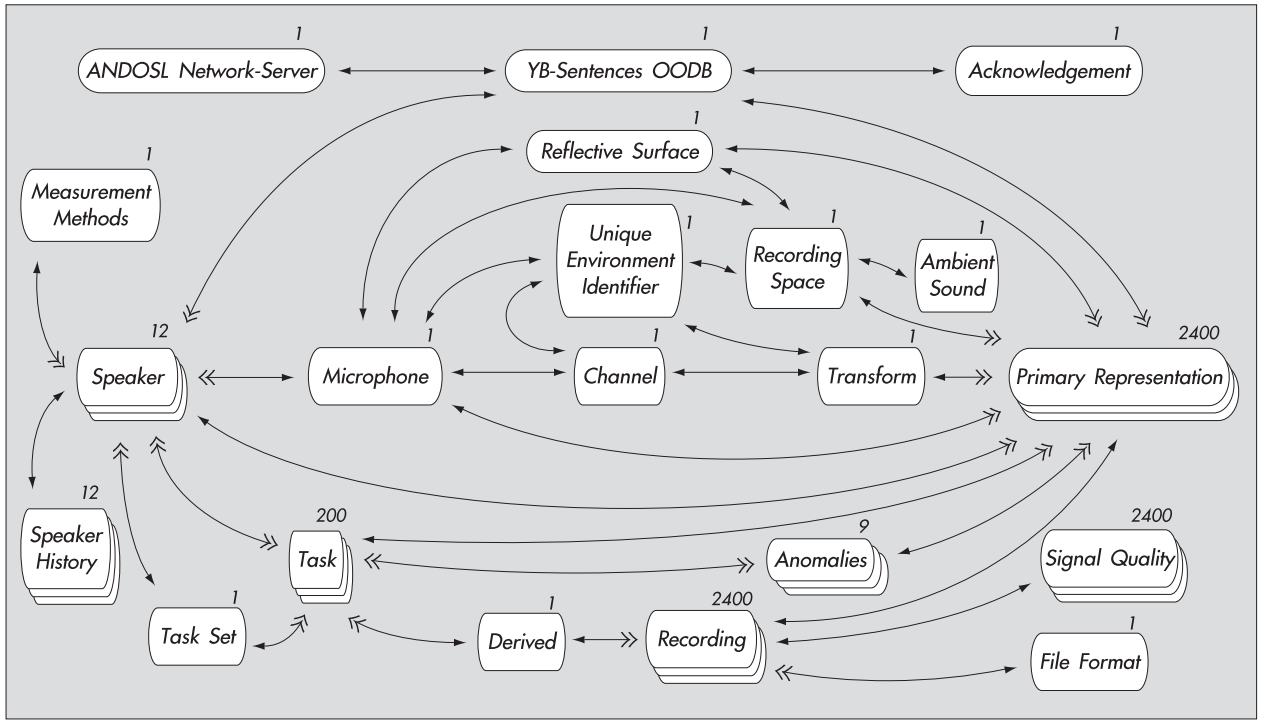


Fig. 22. An object-oriented atemporal model of the data found on one of the ANDOSL CD-ROMs (YB_Sentences). The model is built around 21 different classes; the numeral next to the class indicates how many instances per class were created and exist in the structure. Lines indicate direct relationships: 1-to-1 links (single arrow) and 1-to-many links (double arrows).

oriented division of ANDOSL YB_Sentence material formed a densely linked network structure with 7447 unique objects and 66944 links (number of 1-to-1 links: 34096; number of 1-to-many links: 32848). Only nine `ANOMALY` objects were found since four speakers made identical recording mistakes. The structure including phonemic temporal data occupied 7.5 MB of working memory.

Figure 22 can also be seen as a navigational roadmap for assisting in query formation. Traversal between objects not directly linked to one another is supported via computational links, e.g., a speaker `x` can gain access to all of his/her `RECORDING` objects (200 in this case) via the form `(recordings x)` which returns them as a list.

5.3.5 OO-DBMS Queries

Identical queries to those specified in section 5.3.3 were written as predicate functions in Lisp/CLOS and applied to all ANDOSL speaker objects in the atemporal framework. In the following lambda expressions the variable *speaker* refers to the object being tested. In the first query the speaker is first requested to return all cigarette or smoking episodes as a list of episode objects via a call to *find–episodes–for*. Each episode then calculates it duration and these values are summed to obtain a cumulative value. If the value is larger than 5 years then the predicate returns `T`, otherwise `NIL`. Given this query the search engine returned the matching speaker objects in the framework as a list. If desired, further processing could be applied immediately to these objects.

```
(lambda (speaker)
  (let* ((episodes (find-episodes-for speaker '(smok igarette)))
         (cigarette-durations (duration-of-episodes episodes))
         (cumulative-years (sum-of-elements cigarette-durations)))
    (> cumulative-years 5))))

=> (#<SPEAKER "S111"> #<SPEAKER "S126"> #<SPEAKER "S128">)
```

Query 1 in an OO-DBMS Format.

```
(lambda (speaker)
  (and (> (year_of_birth speaker) 1970)
       (episode-exists-for speaker '(mus piano))
       (loop for primary-representation in (primary-representations speaker)
             for recording = (recording primary-representation)
             for signal-quality = (signal-quality recording)
             do (when (and (= 17 (sentence-number recording))
                           (= 0 (number_of_positive_limits signal-quality))
                           (= 0 (number_of_negative_limits signal-quality)))
                  (return t)))))

=> (#<SPEAKER "S115">)
```

Query 2 in an OO-DBMS Format.

In the second example several conditions need to hold for the predicate expression to return a non–NIL value. The speaker's year–of–birth is first tested to be later than 1970. Then a test is made for at least one episode to exist for speaker where music and piano have existed. Finally, an iteration is performed over all of the speaker's PRIMARY-REPRESENTATIONs by looping. Then access to the PRIMARY-REPRESENTATION's RECORDING and the latter's SIGNAL-QUALITY object is made (see links in Fig. 22). Finally, when the sentence number is equal to 17 and the signal quality is satisfactory, loop returns T (true) and terminates immediately, else after looping over all PRIMARY-REPRESENTATIONs and failing to find a match, NIL is returned instead.

5.3.6 Query Efficiency: RDBMS vs. OO-DBMS

The RDBMS queries were executed with ORACLE running on a dual-processor Sparc–10 system. Query 1 utilised 1.17 cpu-seconds while Query 2 took 9.30 cpu-seconds.

The OO-DBMS tests were run on a 300 MHz PowerPC 750 RISC CPU running Macintosh Common Lisp 4.3b1 with all structure residing in RAM memory. In the OO–DBMS implementation database access time for Query 1 was 1.6 ms. During this time 512 bytes of temporary memory was utilised but was recovered automatically by Lisp's ephemeral garbage collector. Query 2 returned SPEAKER S115 as an object in 4.1 ms and utilised 40 bytes of temporary memory.

It should be noted that queries in QSSDB are not interpreted during database access: Lisp's compiler generates a function from the query which is repeatedly applied by the search engine to seek out matches. Compilation time is included in the above performance figures; slightly better performance could be expected if the queries were precompiled.

The time utilised by the queries as well as the improvement in speed when using the OO–DBMS system are listed in Table 6.

Table 6. Relative difference of an RDBMS vs. OO-DBMS system's performance on identical queries.

Query	*RDBMS Access (sec)*	*OO-DBMS Access (sec)*	*Improvement Ratio*
1	*1.17*	*0.0016*	*730*
2	*9.30*	*0.0041*	*2 300*

Since the execution platforms of the RDBMS and OO–DBMS are different (SPARC vs. Apple; different operating systems; ORACLE vs. MCL Lisp, different metering, etc.) the improvement ratios can be used only as indicators. However, the figures reveal that a vast improvement in performance was observed over the RDBMS model when using the representation framework methodology presented in this thesis.

6 Discussion

In this section research and applications that have relied on QSSDB are covered along with an analysis of the strengths and weaknesses of the system. Finally, future goals and planned development work are mentioned.

6.1 QSSDB Applications

Some of the applications developed by the author at the Helsinki University's Laboratory of Acoustics and Audio Signal Processing which use QuickSig and QSSDB are listed in the following sections. Work is presented according to the fields of speech research covered as well as chronological order as an aid for the reader in understanding the evolution of QSSDB. Other searches of speech data have been processed with QSSDB as well, e.g., for the development of voiced/unvoiced detectors, for time-frequency analysis of speech, and for many other ad hoc experiments where specific contexts of speech have been required.

6.1.1 Speech Recognition

In 1991 work was carried out on event-based recognition and analysis of speech using artificial neural networks (ANN) (Altosaar & Karjalainen, 1991a; Altosaar & Karjalainen, 1991b). Here ANNs were combined with a knowledge-based approach to perform formant analysis and phone recognition for the Finnish language. Training material for ANNs was selected from a prototype speech corpus of 1200 words spoken in isolation by one speaker, of which the majority had been hand-labelled by the author of this thesis over the previous year[3]. Training and evaluation material selection occurred using predicate functions applied to the corpus' phonetic domain representation frameworks. These queries returned as matches the phone objects that matched certain contextual criteria. Method functions were then written for phone units to compute auditory spectrum feature matrices from the acoustic time waveform that they represented. ANNs were then trained and evaluated in the same computational environment.

This work continued and led to the development of time-event neural networks (TENN) which were based on the detection of events as time moments or intervals of interest. The first step utilised a set of parallel ANNs running in time that performed a coarse order of classification according to the transition occurring in speech, i.e., diphone detection. For example, a stop-vowel detector would run through an utterance and provide hypotheses on where such units existed. Each coarse-class detector utilised

[3] This prototype speech corpus marked the beginning of Finnbase as well as the QSSDB system. At this point in time no collection of phonetically annotated Finnish speech existed and its creation was required for speech recognition research. The development environment for this work was based on QuickSig that ran on Symbolics Lisp Machines. The author of this thesis developed the segmentation and hand labelling environment software system used to create Finnbase on both Symbolics and Apple platforms.

speech knowledge to define its specific window-of-interest on the speech signal. Once prominent diphone locations had been found, a second set of specialised networks were applied to these time moments which specified what transition was most likely, e.g., a [p]-[a] diphone. Again, as in previous work, predicate functions were used as queries and yielded phone units that generated their training and evaluation matrices. Training and evaluation, however, took place on an in-house TMS-320 DSP environment controlled by QuickSig.

6.1.2 Speech Analysis

In 1994 the porting of QSSDB to the Apple platform (MCL/CLOS) was begun since Symbolics Lisp Machines (Genera/Flavors) were becoming antiquated. Funding had also been secured to employ phonetician's from the University of Helsinki's Department of Phonetics to segment and hand-label a larger set of Finnish speech: two male speakers each uttering nearly 900 isolated words that covered all diphone transitions of the Finnish language. Once labelled this new data was added to the OO-DBMS of QSSDB and became available for use. Since then the Finnish corpus has grown and evolved: more speakers have been included uttering connected speech and most of the annotation labels originally in Finnbet have been converted to Worldbet. QSSDB along with Finnbet is now in use by the University of Helsinki's Department of Phonetics for speech analysis purposes.

6.1.3 Speech Synthesis

The first application of QSSDB to speech synthesis began with coding phoneme duration rules using ANNs (Karjalainen & Altosaar, 1991). In this work phones in their respective word contexts were used to form an input vector based on symbolic information such as coarse class (e.g., vowel), fine class (e.g., [a]), and quantity (e.g., short or long). This research was expanded to include the study of other elements of prosody, e.g., pitch and loudness—both on a word level and phone level basis (Vainio & Altosaar, 1996a; Vainio & Altosaar, 1998). More recent work has included connected speech as well as the use of morphology in generating training and evaluation material for ANNs.

6.1.4 Speaker Recognition

Experiments in speaker recognition were carried out using the QSSDB system (Altosaar & Meister, 1995). In this series of experiments different spectral representations were used to train ANNs to determine which representation was most robust. Long-term auditory spectra, Fourier spectra, and warped-LPC representations were studied, including different ANN topologies. Database queries only utilised atemporal attributes such as gender and speaker information, since temporal objects were limited to the acoustic signal alone. In terms of hardware base, the ANN algorithms were still executing on an external TMS-320 DSP but by 1996 the author had coded the floating point intensive operations into native PowerPC assembler thus speeding up as well as simplifying—since no external processor

was required—QSSDB signal post-processing operations.

6.2 Strengths and Weaknesses

Modelling using the OO paradigm has been found to be a natural and intuitive way of representing the complex phenomena of speech. Typically, the design of a model is motivated by theory, while its further evolution is induced by empirical results. Being a repetitive looping process of: `design` → `implement` → `test` ↲, *model evolution* relies on the creativity of its designer and new stimuli, e.g., the addition of a new corpus to the system causes a distillation and refinement of existing knowledge-based resources. Therefore, in general, the cost of adding a new corpus is inversely proportional to the number of corpora already covered. The OO paradigm (using CLOS) allows the designer to implement a robust infrastructure on which to create a model, while the interactive environment of LISP and its features, e.g., incremental compiler, backtracing, dynamic binding, and tagged objects, etc., allow for a model's rapid evolution. This leaves the designer with an open system, one whose order and complexity can be increased with a finite amount of effort.

Establishing equivalence relationships between different language-theories and underlying annotation formalisms is outside the scope of the thesis. However, more insight into these difficult problems can be gained by using representation frameworks to measure these equivalences. Comparisons of different theories can be performed by using a unique corpus as source data and representing it with different competing theories within the same representation frameworks. This allows for corpus-wide analyses to be performed which can be used to determine quantitatively the correlates between theories. With suitable classes and methods defined, representation frameworks can be used to establish the relations between different annotation standards. For example, if the user is interested in studying how the acoustic-phonetic units found in the TIMIT corpus correspond to the phonetic units of the Kiel Read Speech Corpus, similar units in defined contexts can first be searched for by identical search functions. Then numerical means can be applied to measure the temporal or spectral characteristics of these units and quantitative analyses used to determine their relationships.

In section 4, the aim of having a compiler and linker void of domain-specific knowledge was emphasised. In the current implementation of QSSDB, atemporal annotations, e.g., those modelled from ANDOSL MAP tasks, still have a unique atemporal compiler and linker. A future goal is to integrate the generation of temporal and atemporal frameworks regardless of the different environments and constraints involved. Since currently ANDOSL data is the only corpus for which an atemporal model has been developed, more corpora, i.e., stimuli, are needed to develop an integrated temporal-atemporal system.

Practical issues such as memory utilisation and processing speed are relevant in development and application work since they affect the effectiveness and viability of a system. For example, corpus data within QSSDB are stored in a persistent object storage system that is an integral part of the QuickSig environment yielding good performance. This reduces the need to cache large objects, e.g., signal arrays, thus freeing up working memory. The ability to dynamically control the detail of a representation framework is a worthy goal. For example, the ability to exclude a certain parallel *di-unit* perspective of the data to save on memory is currently available but other controlling features could be added as well, e.g., deletion of domains not needed in an analysis, etc. Again, more stimuli are called for.

In section 5.3.6 the relative performance of identical queries were compared using a RDBMS and OO-DBMS. The size of corpora that can be processed in QuickSig is not limited by the size of main memory since memory paging can be used, e.g., virtual memory. Scalability was not studied and it remains to be seen whether similar several orders of magnitude gains are possible when very large corpora are processed.

Reading a corpus into QSSDB and generating a representation framework for each utterance is a minor overhead that is incurred only once. For example, with TIMIT data and using the same hardware as mentioned in section 5.3.6, approximately five utterances per second can be processed. Therefore, TIMIT's 6300 utterances can be compiled and linked in about 20 minutes. Once the frameworks are built, they can be either i) used for searching corpora immediately, ii) stored as a Lisp image where time to reuse is a few seconds (as the Lisp environment starts up), iii) stored in an OO-DBMS where reuse is dependent on file system access speed, or iv) stored in a textual format, e.g., XML, which increases portability.

No graphical user interface for automatically generating search functions in Lisp has been developed for QSSDB since this was deemed to be outside the scope of the thesis. A graphical interface for defining search functions could certainly act as an aid to speech engineers, phoneticians, and linguists who are not familiar with a functional programming language.

It should be noted that the goal of this thesis was not to create a portable system that would run on different platforms, but rather to investigate the applicability of the OO paradigm to represent and process speech corpora in a natural and efficient manner.

6.3 Future Development

In the future, speech corpora should supply much improved machine interpretable descriptions of their signals and annotations, as well as a schema allowing for their effective use; currently such items are found to be missing or deficient. A corpus specification model should be integrated with the corpus data since this will add significant value. Hopefully, a unified annotation scheme will be agreed upon in the near future; otherwise, an increasing amount of resources will have to be spent on deciphering corpus formats. In any case, existing corpora will need specification models of their data so that they can be transformed losslessly into a new formalism. To further this we are experimenting with methods to describe corpora as well as representation frameworks in a universal format, e.g., by using XML (World Wide Web Consortium, 2000).

Work has started to include more speech corpora to the QSSDB system, e.g., the large BAS Verbmobil corpora (BAS, 1996), so that scalability issues related to the formalism can be studied more closely.

Inclusion of the visual element in speech will be of significance for future man-machine communication. For this reason plans are being made for extending the QSSDB system to include multi-modal (audio-visual) data on which both numeric and symbolic facial parameters can be annotated. Database queries will then operate on speech and visual units in parallel, e.g., phones, visemes, and facial expression features.

7 Summary

This thesis presented an object-based modelling environment for representing and processing speech corpora. A two-level data abstraction technique for modelling speech signals and their related annotations was described. The first level reinstated missing type and relationship information by developing a corpus specification model and generic representation format. Using a set of orthogonal knowledge-based resources, a second level of abstraction was formed by compiling information into units that exhibit speech knowledge. These units were then placed into a representation framework where a linker tied together related speech units producing a tightly knit framework. Queries that searched out matching contexts in these frameworks returned actual speech units. This enabled further analyses to be performed efficiently with minimal database impedance mismatch.

Five different speech corpora covering four different languages, annotated in five different phonetic alphabets, and consisting of over 17 000 speech utterances were used to develop and test the QSSDB system described in this thesis.

The object-oriented formalism maps well to the modelling and processing tasks encountered in speech. This thesis indicates that by viewing signals and their annotations through an object-oriented perspective, a rich and detailed model of spoken language can be formed. The efficiency of these methods allows large speech databases in multiple languages to be processed and searched using linguistically motivated structures and queries. The object-oriented formalism in turn provides a structure on which to study and better understand the phenomenon of speech.

References

Alarotu N., Lennes M., Altosaar, T., Malm A. & Karjalainen M. (1997). Applications for the Hearing-Impaired: Comprehension of Finnish Text with Phoneme Errors. In Proceedings of EUROSPEECH-97, Rhodes.

Altosaar, T. & Karjalainen M. (1987). An Event-based Approach to Auditory Modeling of Speech Perception. In Proceedings of the ICPhS-87, Tallinn, Estonia.

Altosaar, T. & Karjalainen, M. (1989). A Knowledge-Based Approach to Unlimited Vocabulary Speech Recognition for the Finnish Language. In Proceedings of the European Conference on Speech Communication and Technology. pp. 613-616. Paris, France.

Altosaar, T. & Karjalainen M. (1988). Event-Based Multiple-Resolution Analysis of Speech Signals. In Proceedings of the IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP), New York.

Altosaar, T. & Karjalainen, M. (1991a). Automatic Classification and Formant Analysis of Finnish Vowels Using Neural Networks. In Proceedings of the ICPhS-91, Aix en Provence, France.

Altosaar, T. & Karjalainen, M. (1991b). Event-Based Recognition and Analysis of Speech by Neural Networks. In Proceedings of EUROSPEECH-91. Genova, Italy.

Altosaar, T. & Karjalainen M. (1991c). Vowel Recognition and Analysis Using Neural Networks. In Proceedings of the 1990 Finnish Phonetics Conference. Published in 1991. Oulu, Finland.

Altosaar, T. & Karjalainen M. (1992). Diphone-Based Speech Recognition using Time-Event Neural Networks. In Proceedings of the ICSLP-92, Banff, Canada.

Altosaar, T., Karjalainen M. & Vainio M. (1994). Experiments with an Object-Oriented Database for Finnish: Development and Analyses. In Proceedings of the Finnish Phonetics Symposium, Tampere, Finland.

Altosaar, T. & Meister E. (1995). Speaker Recognition Experiments in Estonian using Multi-Layer Feedforward Neural Nets. In Proceedings of EUROSPEECH-95. Madrid, Spain.

Altosaar, T., Karjalainen M. & Vainio M. (1996a). An Access Method for Different Speech Databases. In Proceedings of the Finnish Phonetics Conference. Joensuu, Finland.

Altosaar, T., Karjalainen M. & Vainio M. (1996b). A Multilingual Phonetic Representation and Analysis System for Different Speech Databases. In Proceedings of the ICSLP-96, Philadelphia.

Altosaar, T., Vainio, M. & Karjalainen M. (1996). Modeling of pitch, loudness, and segmental durations in Finnish using neural networks. Acoustical Society of America Meeting, Honolulu.

Altosaar, T., Karjalainen, M., Vainio, M. & Meister, E. (1998). Finnish and Estonian Speech Applications developed on an Object-Oriented Speech Processing and Database System. In workshop proceedings of First International Conference on Language Resources and Evaluation. Speech Database Development for Central and Eastern European Languages. Paper No. 6. Organised by the BABEL Project, Copernicus No. 1304. Granada, Spain.

Altosaar, T. & Vainio, M. (1998). Forming Generic Models of Speech for Uniform Database Access. In Proceedings of the ICSLP'98 (International Conference on Spoken Language Processing). pp. 3181-3184. Sydney, Australia.

Altosaar, T. & Vainio, M. (1999). Transforming Information in Speech Databases into Knowledge. In Proceedings of the 14th International Congress of Phonetic Sciences. (J. Ohala, Y. Hasegawa, M. Ohala, D. Granville, & A. Bailey eds.). pp. 1737-1740. San Francisco, CA., USA.

Altosaar, T., Millar, B. & Vainio, M. (1999). Relational vs. Object-Oriented Models for Representing Speech: A Comparison Using ANDOSL Data. In Proceedings of the 6th European Conference on Speech Communication and Technology. Vol. 2, pp. 915-918. Budapest, Hungary.

Altosaar, T. & Vainio, M. (2000). Reduced Impedance Mismatch in Speech Database Access. In Proceedings of the ICSLP-2000. vol. 1. pp. 778-781. Beijing, China.

Altosaar, T., Karjalainen, M. & Vainio, M. (2001). Three-Dimensional Modelling of Speech Corpora: Added Value through Visualisation. In Proceedings of the 7th European Conference on Speech Communication and Technology. Alborg, Denmark.

Aoki, A. (2000). Smalltalk Textbook for VisualWorks 1.0, 2.0, and 2.5. URL:

Aoki, A. (2000). Smalltalk Textbook for VisualWorks 1.0, 2.0, and 2.5. URL: http://www.sra.co.jp/people/aoki/SmalltalkTextbook/index.html

Apple Computer, Inc. (2001). QuickDraw-3D. URL: http://www.apple.com.

Awad, M., Kuusela, J. & Ziegler, J. (1996). Object-Oriented Technology for Real-Time Systems: A Practical Approach Using OMT and Fusion, pp. 3-61, Prentice-Hall, Inc. Upper Saddle River, NJ., USA.

Barry, W. & Fourcin, A. (1992). Levels of Labelling. Computer Speech and Language. **6**, No. 1, pp. 1-14.

BAS - Bayerisches Archiv für Sprachsignale. (1996). Verbmobil 2.1 1.02.96. BAS - Institut für Phonetik und Sprachliche Kommunikation - Schellingstr. 3, D 80799 München. URL: http://www.phonetik.uni-muenchen.de/

Bird, S., Day, D., Garofolo, J., Henderson, J., Laprun, C. & Liberman, M. (2000). ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation. In Proceedings of the Second International Conference on Language Resources and Evaluation, pp. 1699-1706, Paris: European Language Resources Association.

Bird, S., Buneman, P. & Tan, W. (2000). Towards a Query Language for Annotation Graphs. In Proceedings of the Second International Conference on Language Resources and Evaluation, pp. 807-814, Paris: European Language Resources Association.

Bird, S. & Liberman, M. (2001). A Formal Framework for Linguistic Annotation. Speech Communication, **33**, pp. 23-60. Elsevier Science Publishers B.V., North-Holland.

Black, A., Taylor, P., Caley, R. & Clark, R. (2000). The Festival Speech Synthesis System. URL: http://www.cstr.ed.ac.uk/projects/festival/festival.html

Blaha, M. & Premerlani, W. (1998). Object-Oriented Modeling and Design for Database Applications. ISBN 0-13-123829-9. Prentice-Hall.

Cassidy, S. & Harrington, J. (1996). EMU: an Enhanced Hierarchical Speech Data Management System. In Proceedings of the Speech Science and Technology Conference. Adelaide, Australia.

System. In Proceedings of the Speech Science and Technology Conference. Adelaide, Australia.

Cassidy, S. (1998). Personal correspondence. Macquarie University. December, 1998.

Cassidy, S. (1999). Personal correspondence. October, 1999.

Chan, D., Fourcin, A., Gibbon, D., Granström, B., Huckvale, M., Kokkinakis, G., Kvale, K., Lamel, L., Lindberg, B., Moreno, A., Mouropoulus, J., Senia, F., Trancoso, I., Veld, C. & Zeiliger, J. (1995). Eurom - a Spoken Language Resource for the EU. In EUROSPEECH '95 Proceedings: ESCA 4th European Conference on Speech Communications and Technology. (J. M. Pardo, E. Enríquez, J. Ortega, J. Ferreiros, J. Macías and F. J. Valverde, eds.). pp. 867-870. Madrid, Spain.

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

Draxler, C. (2001). Personal communication. Institut für Phonetik und Sprachliche Kommunikation, Universität München.

Entropic Ltd. (1999). `esps waves+`. URL: http://www.entropic.com/products&services/esps/esps.html

ESPRIT project 1541. (1987). SAMPA computer readable phonetic alphabet. Documentation available at http://www.phon.ucl.ac.uk/home/sampa/home.htm

European Language Resources Association. (2000). 55 Rue Brillat-Savarin, F-75013, Paris, France. URL: http://www.icp.inpg.fr/ELRA/home.html

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D. & Dahlgren, N. (1993). The DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CD-ROM [TIMIT]. Documentation supporting NIST Speech Disc 1-1.1. NISTIR 4930.

Harrington, J., Cassidy, S., Fletcher, J. & McVeigh, A. (1993). The mu+ system for corpus based speech research. Computer Speech & Language, Vol. 7:4, pp. 305-331.

Hendriks, J. (1990). A Formalism for Speech Database Access. Speech Communication, **9**, pp. 381-

Hendriks, J. (1990). A Formalism for Speech Database Access. Speech Communication, **9**, pp. 381-388. Elsevier Science Publishers B.V., North-Holland.

Hieronymus, J. (1993). ASCII Phonetic Symbols for the World's Languages: Worldbet. Bell Labs Technical Memorandum.

Hughes, J. (1991). Object-Oriented Databases. Prentice-Hall. ISBN 0-13-629874-5.

International Phonetic Association. (1999). Handbook of the International Phonetic Association : A Guide to the Use of the International Phonetic Alphabet. Cambridge University Press.

Itahashi, S. (1986). A Japanese Language Speech Database. In Proceedings of the IEEE International Conference On Acoustics, Speech, and Signal Processing. pp. 321-324. Tokyo, Japan.

Jurafsky, D. and Martin, J.H. (2000). Speech and Language Processing. Prentice Hall, Upper Saddle River, New Jersey.

Karjalainen M., Altosaar, T. & Alku, P. (1988). QuickSig—An Object-Oriented Signal Processing Environment. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. pp. 1682-1685. New York, N.Y., USA.

Karjalainen M. & Altosaar, T. (1988). QuickSig - An Expert Workstation for Numeric and Symbolic Signal Processing. In Proceedings of the EUSIPCO-88, Grenoble, France.

Karjalainen, M., Altosaar, T., Alku P., Lehtinen, L. & Helle, S. (1989). Speech Processing in the Object-Oriented DSP Environment QuickSig. In Proceedings of the European Conference on Speech Communication and Technology. pp. 450-453. Paris, France.

Karjalainen, M. (1990). Object-Oriented Programming: A Case Study of QuickSig. In IEEE Acoustics, Speech and Signal Magazine. pp. 21-31. April, 1990.

Karjalainen, M. Altosaar, T. (1991). Phoneme Duration Rules for Speech Synthesis by Neural Networks. In Proceedings of EUROSPEECH-91. Genova, Italy.

Karjalainen, M., Välimäki, V., Altosaar, T., & Helle, S. (1992). The QuickSig System and its Computer Music Applications. In Proceedings of the International Computer Music Conference. pp.

Computer Music Applications. In Proceedings of the International Computer Music Conference. pp. 390-391. San Jose, California, USA.

Karjalainen M. & Altosaar, T. (1992). The QS-Nets Neural Computation Environment. In Proceedings of STeP-92 (Finnish Artificial Intelligence Symposium), Espoo, Finland.

Karjalainen, M. & Altosaar, T. (1993). An Object-Oriented Database for Speech Processing. Proceedings of the European Conference on Speech Communication and Technology. pp. 183-186. Berlin, Germany.

Karjalainen M., Boda P., Somervuo P. & Altosaar, T. (1997). Applications for the Hearing Impaired: Evaluation of Finnish Phoneme Recognition Methods. In Proceedings of EUROSPEECH-97. Rhodes.

Karjalainen, M., Altosaar, T. &Vainio, M. (1998a). Speech Synthesis using Warped Linear Prediction and Neural Networks. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. pp. 877-880. Seattle, Washington, USA.

Karjalainen M. & Altosaar, T. (1998a). An Automated Labeling Tool for Speech Database Work. In Proceedings of Norsig-98. pp. 113-116. Denmark.

Karjalainen M. & Altosaar, T. (1998b). A Tool for Automatic Labelling of Finnish Speech. In Proceedings of the Finnic Phonetic Symposium. pp. 220-225. Pärnu, Estonia.

Karjalainen M., Altosaar, T., & Vainio M. (1998b). Finnish Speech Synthesis using Warped Linear Prediction and Neural Networks. In Proceedings of the Finnic Phonetic Symposium. pp. 205-212. Pärnu, Estonia.

Karjalainen, M., Altosaar, T. & Huttunen, M. (1998). An Efficient Labeling Tool for the QuickSig Speech Database. In Proceedings of the ICSLP'98 (International Conference on Spoken Language Processing). pp. 1535-1538. Sydney, Australia.

Keene, S. (1989). Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS. Addison-Wesley.

Kelly, J. & Local, J. (1989). Doing Phonology. Manchester University Press, Manchester, UK.

Kelly, J. & Local, J. (1989). Doing Phonology. Manchester University Press, Manchester, UK.

Kohler, K. (1998). Lectures and workshop on phonetics. Department of Phonetics, University of Helsinki. March 24-27, 1998.

Kopec, G. (1980). The Representation of Discrete-Time Signals and Systems in Programs. Ph.D. dissertation. Massachusetts Institute of Technology. Cambridge, Mass., USA.

Kopec, G. (1985). The Signal Representation Language SRL. IEEE Transactions on Acoustics, Speech, and Signal Processing. p. 921. ASSP-33, No. 4.

Laine U.K. & Altosaar, T. (1990). An Orthogonal Set of Frequency and Amplitude Modulated (FAM) Functions for Variable Resolution Signal Analysis. In Proceedings of IEEE ICASSP-90, Albuquerque, New Mexico.

Laine U.K., Karjalainen M. & Altosaar, T. (1991). Time-Frequency and Multiple-Resolution Representations in Auditory Modeling. In Proceedings of IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY.

Laine U.K., Karjalainen M. & Altosaar, T. (1994). Warped Linear Prediction (WLP) in Speech and Audio Processing. In Proceedings of IEEE ICASSP-94, Adelaide, Australia.

Linguistic Data Consortium. (2000). University of Pennsylvania, 3615 Market St., Philadelphia PA 19104-2608, USA. URL: http://www.ldc.upenn.edu/

Meister, E. & Eek, A. (1999). Estonian Phonetic Database. In Babel: A Multi-Language Database. EU Copernicus Programme. Project No. 1304. Tallinn, Estonia.

Meister, E., Eek, A., Altosaar, T. & Vainio, M. (1999). The Estonian Phonetic Database in the QuickSig Object-Oriented Environment. In Proceedings of the International Workshop on Computational Linguistics and its Applications - Dialogue'99. vol.2, pp.347-350, Tarusa, Russia.

Meister, E., Eek, A., Altosaar, T. & Vainio, M. (2000). Object-oriented access to the Estonian phonetic database. - In Proceedings of the 2nd International Conference on Language Resources and Evaluation. vol.1, pp. 269-272. Athens, Greece.

Millar, J. (1992). The Description of Spoken Language. In Proceedings of SST-92. pp. 80-85. Brisbane, Australia.

Millar, J.,Vonwiller, J., Harrington, J. & Dermody, P. (1994). The Australian National Database Of Spoken Language. In Proceedings of IEEE International Conference On Acoustics, Speech, and Signal Processing. pp.97-100. Adelaide, Australia.

Millar, J., Harrington, J. & Vonwiller, J. (1997). Spoken Language Resources for Australian Speech Technology. Journal of Electrical and Electronic Engineering Australia, Vol.17:1, pp.13-23.

Millar, J. (1998). A structure for comprehensive spoken language description. In Proceedings of ICLRE. pp.1303-1308. Granada, Spain.

Myers, C. (1986). Signal Representation for Symbolic and Numerical Processing. RLE Technical Report 521. Massachusetts Institute of Technology. Cambridge, Mass., USA.

Olivés, J.-L., Sams, M., Kulju, J., Seppälä, O., Karjalainen, M., Altosaar, T., Lemmetty, S., Töyrä, K. & Vainio, M. (1999). Towards A High Quality Finnish Talking Head. In Proceedings of the 1999 IEEE International Workshop on Multimedia Signal Processing. pp. 433-437. Denmark.

Oppenheim, A. & Hamid, S. (eds.). (1992). Symbolic and Knowledge-Based Signal Processing. Prentice-Hall, Englewood Cliffs, NJ., USA.

Preiss, B. (1999). Data Structures and Algorithms with Object-Oriented Design Patterns in Java. University of Waterloo. URL: http://dictator.uwaterloo.ca/Bruno.Preiss/books/opus5/main.html

Simpson, A., Kohler, K., & Rettstadt, T. (eds.). (1997). The Kiel Corpus of Read/Spontaneous Speech - Acoustic data base, processing tools and analysis results. Arbeitsberichte (AIPUK) Nr. 32. Universität Kiel, Kiel, Germany.

Stajano, F. (1998). A Gentle Introduction to Relational and Object Oriented Databases. ORL Technical Report TR-98-2. URL: http://www.uk.research.att.com/~fms/db/

Steele, G. (1990). Common Lisp. 2nd edition. Digital Equipment Corporation. Digital Press. Bedford, Mass., USA.

Mass., USA.

Stroustrup, B. (1997). The C++ Programming Language. Addison Wesley Longman, Reading, MA, USA.

Taylor, P., Black, A. & Caley, R. (2001). Heterogeneous Relation Graphs as a Formalism for Representing Linguistic Information. Speech Communication, **33**, pp. 153-174. Elsevier Science Publishers B.V., North-Holland.

The Mathworks. (2001). MATLAB. URL: http://www.mathworks.com/

Vainio M., Altosaar, T., Karjalainen M. & Iivonen A. (1995). Experiments with an Object-Oriented Speech Database. In Proceedings of the ICPhS-95. Stockholm.

Vainio, M. & Altosaar, T. (1996a). Pitch, Loudness, and Segmental Duration Correlates: Towards a Model for the Phonetic Aspects of Finnish Prosody. In Proceedings of the ICSLP 1996. Philadelphia, USA.

Vainio M. & Altosaar, T. (1996b). Pitch, Loudness, and Segmental Duration Correlates in Finnish Prosody. In Proceedings of Nordic Prosody VII. Joensuu, Finland 1996.

Vainio M., Aulanko R., Altosaar, T. & Karjalainen M. (1997). Modeling Finnish Microprosody for Speech Synthesis. In Proceedings of the ESCA Workshop on Intonation Theory and Applications, pp. 309-312. Athens.

Vainio M., Altosaar, T., Karjalainen M. & Aulanko R. (1998). Modelling Finnish microprosody with neural networks. In Proceedings of the Finnic Phonetic Symposium. pp. 199-204. Pärnu, Estonia.

Vainio, M. & Altosaar, T. (1998). Modeling the Microprosody of Pitch and Loudness for Speech Synthesis with Neural Networks. In Proceedings of the ICSLP'98 (International Conference on Spoken Language Processing). pp. 1931-1934. Sydney, Australia.

Vainio, M., Altosaar, T., Karjalainen, M., Aulanko, R., &Werner, S. (1999). Neural Network Models for Finnish Prosody. In Proceedings of the ICPhS 1999. Vol. 3, pp. 2347-2350. San Francisco.

Vainio, M. & Altosaar, T. (2000). The Importance of Morphological Information for Finnish Speech Synthesis. In Proceedings of the ICSLP-2000. Beijing, China.

Vonwiller, J., Rogers, I., Cleirigh, C., & Lewis, W. (1996) Speaker and Material Selection for the Australian National Database of Spoken Language. Journal of Quantitative Linguistics Vol. 2:3, pp.177-211.

Wolfram Research, Inc. (2001). Mathematica. URL: http://www.wolfram.com/

World Wide Web Consortium (W3C). (2000). Extensible Markup Language (XML). URL: http://www.w3.org/XML/

Zue, V., Cyphers, S., Kassel, R., Kaufman, D., Leung, H., Randolph, M., Seneff, S., Unverferth, J. & Wilson, T. (1986). The Development of the MIT Lisp-Machine Based Speech Research Workstation. In Proceedings of the 1986 IEEE International Conference on Acoustics, Speech, and Signal Processing. pp. 329-332. Tokyo, Japan.

HELSINKI UNIVERSITY OF TECHNOLOGY

LABORATORY OF ACOUSTICS AND AUDIO SIGNAL PROCESSING

34 V. Välimäki: Fractional Delay Waveguide Modeling of Acoustic Tubes. 1994

35 T. I. Laakso, V. Välimäki, M. Karjalainen, U. K. Laine: Crushing the Delay—Tools for Fractional Delay Filter Design. 1994

36 J. Backman, J. Huopaniemi, M. Rahkila (toim.): Tilakuuleminen ja auralisaatio. Akustiikan seminaari 1995

37 V. Välimäki: Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters. 1995

38 T. Lahti: Akustinen mittaustekniikka. 2. korjattu painos. 1997

39 M. Karjalainen, V. Välimäki (toim.): Akustisten järjestelmien diskreettiaikaiset mallit ja soittimien mallipohjainen äänisynteesi. Äänenkäsittelyn seminaari 1995

40 M. Karjalainen (toim.): Aktiivinen äänenhallinta. Akustiikan seminaari 1996

41 M. Karjalainen (toim.): Digitaaliaudion signaalinkäsittelymenetelmiä. Äänenkäsittelyn seminaari 1996

42 M. Huotilainen, J. Sinkkonen, H. Tiitinen, R. J. Ilmoniemi, E. Pekkonen, L. Parkkonen, R. Näätänen: Intensity Representation in the Human Auditory Cortex. 1997

43 M. Huotilainen: Magnetoencephalography in the Study of Cortical Auditory Processing. 1997

44 M. Karjalainen, J. Backman, L. Savioja (toim.): Akustiikan laskennallinen mallintaminen. Akustiikan seminaari 1997

45 V. Välimäki, M. Karjalainen (toim.): Aktiivisen melunvaimennuksen signaalinkäsittelyalgoritmit. Äänenkäsittelyn seminaari 1997

46 T. Tolonen: Model-Based Analysis and Resynthesis of Acoustic Guitar Tones. 1998

47 H. Järveläinen, M. Karjalainen, P. Maijala, K. Saarinen, J. Tanttari: Työkoneiden ohjaamomelun häiritsevyys ja sen vähentäminen. 1998

HELSINKI UNIVERSITY OF TECHNOLOGY
LABORATORY OF ACOUSTICS AND AUDIO SIGNAL PROCESSING

48 T. Tolonen, V. Välimäki, M. Karjalainen: Evaluation of Modern Sound Synthesis Methods. 1998

49 M. Karjalainen, V. Välimäki (toim.): Äänenlaatu. Akustiikan seminaari 1998

50 V. Välimäki , M. Karjalainen (toim.): Signaalinkäsittely audiotekniikassa, akustiikassa musiikissa. Äänenkäsittelyn seminaari 1998

51 M. Karjalainen: Kommunikaatioakustiikka. 1998

52 M. Karjalainen (toim.): Kuulon mallit ja niiden sovellutukset. Akustiikan seminaari 1999

53 Huopaniemi, Jyri: Virtual Acoustics And 3-D Sound In Multimedia Signal Processing. 1999

54 Bank, Balázs: Physics-Based Sound Synthesis of the Piano. 2000

55 Tolonen, Tero: Object-Based Sound Source Modeling. 2000

56 Hongisto, Valtteri: Airborne Sound Insulation of Wall Structures — Measurement And Prediction Methods. 2000

57 Zacharov, Nick: Perceptual Studies On Spatial Sound Reproduction Systems. 2000

58 Varho, Susanna: New Linear Predictive Methods For Digital Speech Processing. 2001

59 Pulkki, Ville; Karjalainen, Matti: Localization Of Amplitude-Panned Virtual Sources. 2001

60 Härmä, Aki: Linear Predictive Coding With Modified Filter Structures. 2001

61 Härmä, Aki: Frequency-Warped Autoregressive Modeling And Filtering. 2001

62 Pulkki. Ville: Spatial Sound Generation and Perception by Amplitude Panning Techniques. 2001

ISBN 951-22-5622-3
ISSN 1456-6303