LINGUISTIC CONSTRAINTS ON SPEECH RECOGNITION

by

DAVID LINDSAY

A Thesis submitted to the University of Keele
for the degree of Doctor of Philosophy

Department of Communication and Neuroscience

1984

ABSTRACT

This thesis is a study of the influence of linguistic constraints on automatic speech recognition by computer. The strategy is to constrain the recognition process by knowledge about the pattern to be recognised, in this case the intonation system of British English.

The categorical nature of the perception of pitch movement corresponding to nuclear syllable intonation is demonstrated. It is shown that Halliday's system of five primary tones is appropriate and applicable to automatic intonation analysis.

A computer analysis system was constructed which uses dynamic programming time warping to compare fundamental frequency patterns. The analysis is constrained by an intonation tone group structure grammar. The grammar consists of context-free rewrite rules and a lexicon of intonation templates. The analysis system comprises a rule translator, a syntax-directed analyser, dynamic programming fundamental frequency contour matcher, and a speech preprocessor.

The system was used in nuclear tone analysis and classification experiments for speaker dependent and independent tone recognition, and for connected utterance analysis over the complete tone group. The results show that a limited prosodics-only speech recogniser is practical.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr W A Ainsworth, for his help and support throughout this project.

The work reported here was carried out in the Communication and Neuroscience Department, University of Keele. I thank all those who helped or acted as experimental subjects in the course of the project.

This work was financially supported by a Medical Research Council research studentship, and latterly by a Research Fellowship funded by the Joint Speech Research Unit, Cheltenham, for research on prosodic aids for speech recognition.

This work would not have been possible without the support and encouragement of my parents.

CONTENTS

CHAPTER 1

A COMPOSITE STRATEGY ON THE PROBLEM OF AUTOMATIC SPEECH RECOGNITION

Speech recognition by machine

The most important means of communication between people is speech - the primary manifestation of language. Until very recently, the ability to speak, the ability to use both the physical apparatus and the control mechanisms involved, has been a unique and identifying trait. Speech is an important feature of human society: man has been defined as the talking animal by Aristotle; and the facility of speech has even been described as an evolutionary trait (Lieberman 1968; Stevens 1975; Lieberman 1981). With the introduction of the stored instruction program devices in the form of digital computers, considerable effort has been spent into allowing these machines to use that distinguishing quality of language.

We can view language as a means of communication. One important model of this function of language derives from the mathematical theory of communication (Shannon and Weaver 1949), making distinct the processes of transmission and reception, and in turn distinguishes these from the communication channel. Building further on this, we may look on sound as a channel for speech, and also on language as a channel, itself a code, for meaning (Lamb 1966; Lockwood 1971). Just as the communications engineer exploits his knowledge of the communication channel he is dealing with to the full, so we can use our knowledge of the language code to enable improved machine analysis and synthesis of speech.

One of the problems in dealing with a communication system lies in the encoding and subsequent decoding processes, the intermediary stages between the information in human form and the transmission by the channel. For people using this part of the process it is the familiar talking and listening of everyday activity; the machine use of speech will entail the achievement of the operationally similar activities of speech synthesis and speech analysis. It is often difficult to realise the scale of these problems: we use speech so fluently and thoughtlessly that we take language for granted. In order to get a computer to analyse/synthesise (listen/talk) one strategy is to mimic in some way just such elements and processes that are normally implicit in human speech.

We need to know just what is used in speech communication and how these components are used. However, there is no readily available body of knowledge; it is not a trivial task to acquire this knowledge - this is one of the goals of linguistics. Being able to construct machines that use speech in a way in some sense analogous to human performance can be considered a form of modelling human communication: in order to model any process we need knowledge of the essential components and their interrelation. The composite strategy of this work is to gain knowledge of one aspect of language, and to use that knowledge in automatic speech recognition.

As well as the acquisition of knowledge, there is the consideration of social and economic advantages of enabling man-machine communication by the speech channel. Computers are already important in many peoples lives, and are destined to play an even more central role in the future. The convenient and efficient use of computers will be enhanced by the ease in which we communicate with them. For example, the common form of communication with computers at the present is via a Visual Display Unit. Nearly all these units could be replaced with voice-mode interfaces, with great advantage. For a full list of social, economic, technological, and military benefits see Martin (1980), and Lea and Shoup (1980). Speech is the natural mode of communication for people: it is a rich and flexible communication system.

At first it was difficult for researchers in the field to recognise the scale of the problems involved. The first successes of automatic speech recognition by computer were confined to isolated word recognition schemes. The paradigm here is the recognition of the ten digits spoken by one speaker, with training of the machine on several runs of the utterances (Davis *et al.* 1952). The method was usually the parameterisation (formant amplitude ratios, or of zero-crossing counts) of an utterance and subsequent matching against a list of stored templates. Simply, the match with the highest score indicated what had been spoken. Isolated word recognition was far more removed from continuous speech recognition than was realised at first, and these methods failed badly when presented with natural connected word utterances.

It was to be expected that such a seriously deficient model of language could not be expected to deal with connected utterances. The differences between the two forms of speech lies mainly in coarticulation effects and with the overlaid realisation of prosodic structure (the reducing of non-stressed vowels and syllable-timing considerations: Crystal and Quirk 1964). There *are* discrete units present in speech, but they are of perception. The mistake was to assume that there is a one-to-one mapping between two representations of speech: perceptual and acoustic.

The approaches attempted at this point consisted of articulating the identification process into a phonemic transcription stage and a symbol-string identification procedure operating on the results of this first stage. Such an approach is an echo of the linguistic theories of the early Structuralists such as de Saussure (1959) and of more recent work by Martinet (1949) which postulates a double articulation of language. This accords with the naive conception of language as being composed of letters (*perhaps* equivalent to an orthographic transcription of phonemes) and words (*perhaps* equivalent to an orthographic transcription of form-classes). Again a one-to-one mapping was tried; this time between a description of language (in two layers) and the processes required to recognise what is described by that model. On the information contained on the phonetic transcription level, often there could be many candidate patterns, many lexical hypotheses formed. The phonological rules of any grammar are mainly concerned with the discarding of information from (over-specified) lexical entries (the *lexical redundancy* concept of transformational generative; Chomsky 1965, p164-170). The way to choose between lexical hypotheses is by using information about what word sequences are possible (syntax), what sequences make sense (semantics) and what sequences are plausible in a situation (pragmatics). Information about what is expected constrains the recognition process.

Automatic speech recognition (ASR) attempts continued from here with projects such as the Advanced Research Projects Agency Speech Understanding Research (ARPA SUR) projects, which displayed a far greater degree of linguistic sophistication. A review of this work is given in section 2.2.

The progress of automatic speech recognition has been through increasingly more appropriate models of language, combined with a similarly increasing ability to make use of them on computer systems. The type of linguistic constraint of interest in this research is knowledge of prosodic structure, especially the intonation system of connected speech utterances. The motivation for choosing this area comes from several sources:

1. Any additional constraint on the speech recognition process can only improve its performance with regard to its recognition accuracy. We must distinguish between algorithm and implementation here. The question of efficiency (approximately equated with number of instructions per second of speech, for digital computation) is not considered here to be relevant as a limiting factor. For a research project, any correct algorithm can be made to work fast enough when hosted on a fast enough machine. Considerations like that, and others like the current availability of special-purpose hardware and program code optimisation, reduces much of the criticism of a purely research-orientated project; the primary task here is to develop a suitable recognition strategy.

   Also, for a rich enough set of possible input utterances, it will eventually be found that the additional complexity of recognition by a highly constrained system is less than for a less-constrained system. A reduction in hypothesis search space due to constraint by a more powerful model will only be evident when the differential performance between using that space and using the less-constrained space is large enough. One comment on the HARPY (Newel *et al.* 1971) system was that its list grammar approach to modelling possible word-sequences was probably the best for such finite grammars (Lea and Shoup 1980). So, although the extensive use of information about prosody is a constraint in automatic speech recognition which has not been tried before, there is no reason for expecting a lessening in the efficiency and, perhaps, every reason for expecting an increase in performance because of it.

2. The additional complexities in connected speech utterance recognition as opposed to isolated word recognition were not due

only to coarticulation effects between neighbouring segments. There are also non-segmental features present in speech that greatly affect the surface structure of speech on which a speech recogniser operates. This fact has been recognised in the attention paid to vowel reduction as a consequence of nuclear stress placement in the Sound Pattern of English model (Chomsky and Halle 1968 generally and also chapter 3 on non-segmental phonetics and phonology).

This work will take the following course. The remainder of this chapter and the next will consist of an introduction to the background of present-day research on the various topics of prosodic phonology, pattern recognition by machine and automatic speech recognition:

1. The first area to be introduced will be the place of knowledge sources in automatic speech recognition. The need for an explicit delineation of the mechanisms, processes and constraints has been mentioned already. Within the modelling of a speech recognition process the part played by such knowledge sources must be seen as a separate role, distinguished from the other tasks occurring within a complex computer system.

2. Then the position of phonology and how it can be used in automatic speech recognition is discussed. It is important to distinguish between phonetic representation and phonemic system. It is transition from registering a phonic event in speech to the recognition of elements of a language system that lifts computers from being measuring tools to their use as recognition machines.

Aims of this research

This chapter has outlined several problems in the field of speech science. From the beginnings of automatic speech recognition there has been a need for knowledge, implicit or explicit, to constrain the recognition system and increase performance. With the shift of critical limitation from hardware to software in recent years emphasis was placed on the explicit use of knowledge sources as descriptions of objects to be recognised. In turn, it was seen that for speech recognition linguistic information could be used as such a knowledge source. At least since Bloomfield (1933) the *explicit* goal of linguistics has been to produce grammars of languages. A grammar is an observationally accurate, exhaustive and simple description (*Conditions of Adequacy*, Chomsky 1957, p47-60) of the pattern underlying the phenomena of speech. As such, a grammar appears to be an ideal candidate for bringing constraining knowledge to automatic speech recognition systems.

However, although grammar-construction is the avowed aim of linguistics, there is yet to be a complete grammar of a natural language. Only fragments of grammars have been produced, usually for the sake of discussing how it should be done - something which is also a problem of the highest order in its own right. Also, there are areas of linguistic knowledge that are not well described: the realm of prosody is one. There is no consensus of theoretical approach, as well as no exhaustive description amenable to formalisation. The aim of this research is to investigate just how to use a linguistic knowledge source in speech recognition. By choosing an ill-defined area of linguistic knowledge in which to work, prosody, we can set up a secondary aim. We will need to investigate that area in order to achieve a description which will be useful in subsequent recognition. So there are two main areas of work here, forming a composite approach:

1. Constructing a description, a grammar, of British English prosody. The programme concerning the description of British English prosody will be as follows. Chapter 3 describes the theoretical justification for our attempt at description. The linguistic status of prosodic effects is argued for and the notion of separating linguistic from paralinguistic features are defended. In section 3.6 a grammar of British English intonation will be presented using a formalisation that can be used by a digital computer system. Preprocessing needs to be discussed as it is pertinent to any acoustic based analysis of prosody - this will be covered in chapter 5. Finally, in chapter 6 the analysis system will be described together with its performance on intonation analysis using various models.

2. To use that model of the British English intonation system to constrain the operation of a purpose-built recognition system. The syntax-directed analysis (SDA) system was constructed as part of this project. It is a prosodics-only speech recogniser that is constrained by a separate description of intonation phenomena. It consists of several components:

   Preprocessor. The relevant prosodic features have to be extracted from speech before analysis. This is described under the heading of acoustic-phonetic considerations, section 5.2.
   Rule translator. The linguistic description is presented to the SDA system as a set of rules that are compiled into an

intermediate representation before the analysis proper. This process is described in section 4.4.

Linker. The intermediate representation must be consolidated into a form usable by a particular program run. Section 4.5.

Template matcher. The acoustic patterns used as templates are matched onto sections of the input speech. The algorithm used is described in section 5.4, the implementation in section 4.3.

Analyser. The operation of the analyser is described in section 6.1. The implementation details can be gathered from section 4.1, 4.2 and the appendices containing program source code.

Before the project is described in detail, however, there is a review of past work on the subjects of intonation theories and of linguistic constraints on ASR.

CHAPTER 2

REVIEW OF PREVIOUS WORK

2.1 Introduction to the literature

This chapter is a review of past work on the research topic. Although automatic speech recognition has a short history - the main lines of research were opened up with the increasing availability of the computer - it has a wealth of literature which has kept pace with both the increase in both depth and breadth of the subject. No survey of this size could possibly do justice to even a small area of what speech science has become: the interdisciplinary nature of research encompasses ideas, techniques and devices from disciplines as diverse as signal processing, computer science, psychophysics, neurology and the linguistic sciences. Approaches are as numerous as the researchers in the field. There is little to be gained, then, by attempting to survey the whole a such an ill-defined field, and little to be lost from focusing attention on the three constituent areas of the present topic:

1. Current automatic speech recognition work. In section 2.2 an outline is presented on the field of automatic speech recognition in order to give some idea of what any knowledge source is to be used in. Again no mention is made of what forces shape the state of the art, and emphasis has been taken away from the interdisciplinary nature of automatic speech recognition. Some of the most instructive projects are the ARPA SUR projects of the early 1970's, discussed below. This is not to deny that there is no other work of note, rather that these projects were the first major attempts at the explicit and systematic use of knowledge sources in constraining a recognition process for speech: the history of the projects illustrate that attempt and would be of interest just for that alone.

   Various other approaches in vogue at the moment include a range of systems: from the LOGOS system - a development of dynamic programming techniques applied (conceptually) to a succession of isolated word templates (Bridle 1973 for the algorithm; J Peckham *et al.* 1982 for the implementation) - to the many English parsing schemes implemented on computer systems (a good survey of text linguistics is de Beaugrande and Dressler 1981; the classical statement of Augmented Transition Network development is Woods 1970; see also Winograd 1983) - which would be applicable to automatic speech recognition. The ARPA projects are doubly instructive because they provide an example of the 'mixing of levels' problem from linguistics, as it applies to automatic speech recognition. What was a methodological crisis for linguistics in the mid 20th Century has become an issue just being felt in Artificial Intelligence work. The direct relevance to this thesis lies in the fact that there is a similar pressure to-day to account for prosody in terms of pragmatics (and possibly even non-verbal behaviour) and not as a (collection of) syntactic mechanisms.

2. Theoretical advances in the linguistic sciences. In section 2.3 a similar outline of current views on the intonation system is given, with no attempt at either a diachronic study of how suprasegmental features have been used in speech, or a study of how differing theoretical positions within linguistics itself have influenced the first-order work of describing the intonation system.

3. The use of prosody in driving automatic speech recognition systems is discussed in section 2.4. The topic of using prosodic information within a automatic speech recognition system is not very well served in the literature. Explicit mention of knowledge sources are

comparatively recent innovation in automatic speech recognition work, and the notion of prosodic knowledge in automatic speech recognition even scarcer (with one or two notable exceptions). There has been considerable amount of effort put into the rather tangential problem of the computer manipulation of grammar, especially transformational generative systems. This includes Marcus (1981), Friedman (1976) - Winograd (1983) gives a good survey.

2.2 The current state of automatic speech recognition

The field of continuous automatic speech recognition is still dominated by the speech understanding research efforts funded by the Advanced Projects Agency of the USA Department of Defence (ARPA SUR) of the late 1970s. These were a group of related projects undertaken by several universities for a period of about about five years. The philosophy behind the projects was to give a spur to work in the speech-related problems of man-machine communication, applying a large amount of expertise, manpower and funds to what was perceived to be an engineering problem. The project report is given by Newel *et al.* (1973) and there have been several excellent and extensive reviews since then (see Klatt 1977; Lea and Shoup 1980). This effort was the largest ever in automatic speech recognition and provided a significant contribution and introduced many new approaches to the repertory of expertise (new, at least, to non-linguists). These projects focused attention on the problems of integrating knowledge sources in pattern recognition systems. As this thesis is presented as a solution to one such problem it is instructive to look at the range and type of knowledge source used. A summary of the knowledge sources used in each of the main systems is given in tables 2.1 to 2.4.

Perhaps the most trenchant criticism is given by Klatt (1977) on the overall design of the project. The project goals could be met in more than one way: a 'solution' may lie outside the spirit of the original specification. In such large and complex systems there will be a constant interplay of efficiency trade-offs; but to specify 'understanding', which is verified by action, instead of 'recognition', which is verified by the accuracy of transcription, was to confuse the roles of pragmatics and syntax and so to lessen the achievement. That point will be discussed below, but there also were many more mundane features that were treated as being 'optional' to a system, to be left out on the grounds of performance. For example, it is still unclear as to how an increase in the grammatical branching factor, and the trade-off between that factor and lexicon size affect the overall recognition score (assuming that we *can* interpret an action as correct recognition). This is a very important point in any discussion of language recognition. There is the anecdote about the chess-playing computer system that announces 'mate in two' after it's opponent coughs: a good illustration of the dangers inherent on short-circuiting syntax to depend on pragmatics to recognise what is said.

A criticism that can be made of the overall strategy is simply that the goals were far too difficult, forcing the researchers to simplify around the problem, avoiding the original task. We can take the HEARSAY II system and its task of document retrieval as an example here. When we take the classical speech recognition process as defined we see that HEARSAY II was very ambitious indeed: after the recognition components there is a hierarchy of subsequent components, each one of substantial complexity. Such a system, if successful, would be most impressive. One of the results of this was a 'mixing of

system SDC

grammar type context-free, minor semantic constraints on number category, case restriction, explicit verb required.

parser implementation rule interpreter, multiprocessing of alternative parse paths, dynamic control of grammatical processes

phonological component generative rules

phonological component implementation automatic readjustment, one example rule is that of vowel reduction, particularised for /i/ (stress: 1)

|REDUCE2 IX=IH:1, .

branching factor 105.

lexicon size 1000.

score (understanding task) 24%.

Table 2.1 *Grammar Used in Constraining ASR (1)*

system BBN Hwim

grammar type pragmatic, augmented transition network grammar (ATN) (Woods 1970). Combines syntactic, semantic, pragmatic, prosodic information. Static embedding of data within the lexicon. (Note in text.)

parser implementation three processes involved

a. judges grammaticality, equivalent to the transition between network nodes in parsing.
b. prediction of possible extensions to hypothesised construction, equivalent to listing set of possible transitions.
c. builds up formal representation of utterance, equivalent to transcribing 'contents' of each node in network visited in transition.

phonological component Dictionary Expansion component comprising dictionary of baseform pronunciation and several sets of phonological rules. The 1976 system gave

baseform (1138 examples)
regular inflections (giving 1363 entries)
giving 1097 entries (after redundancy pruning)
multiple baseform definitions (increasing number to 1789)
within-word rules (giving 3371)
across word rules (resulting in 8642 entries)

phonological component implementation recompilation of network prior to runtime.

branching factor 195.

score (understanding task) 44%.

Table 2.2 *Grammars Used in Constraining ASR (2)*

system CMU Harpy

grammar type rewrite rules compiled into static directed graph representation as one of many knowledge sources. Two techniques were used: substitution (following rewrite rules) and pointer replacement (category, case restriction, explicit verb required).

parser implementation construction of optimal sequence of phones (network nodes) constituting legal paths with high acoustic match probabilities. 'Beam search' of best path with near-miss alternative.

phonological component a multi-source handling, in keeping with the knowledge source philosophy. A pronuciation dictionary for the grammar terminal symbols is combined with juncture rules and 'phrase templates'. The juncture rules were handled in the form of graph rewrite rules, which is just as well, since they would have to operate on the Harpy directed graph anyway.

phonological component implementation operates in compilation of network, prior to runtime.

branching factor 33.

lexicon size 1011.

score (understanding task) 95%.

Table 2.3 *Grammars Used in Constraining ASR (3)*

system CMU Hearsay II

grammar type constraints on word-pairs that are time-adjacent. Information is easily handled here by making use of redundant non-terminal symbols in the grammar.

parser implementation separate knowledge sources interacting on a global blackboard, each knowledge source module being data-driven.

phonological component controlled by knowledge source which generates hypotheses for syllable classes, by parsing out from syllable nuclei. This parsing is controlled by a probabilistic grammar of rules and probability

phonological component implementation generation of several alternative hypotheses, used by knowledge sources to generate word hypotheses. Evaluation of these is performed by a miniature Harpy-like network representation of each lexical entry.

branching factor 33, 46.

lexicon size 1011.

score (understanding task) 91%, 74%.

Table 2.4 *Grammars Used in Constraining ASR - ARPA SUR (4)*

levels' in the system: the way the projects were set up, and the currently available knowledge at the time both conspired against a proper treatment of linguistic constraints in the speech recognition process. For the purposes of comparison, the continuous speech recognition problem can be usefully considered as the attempt to map the speech stream onto a sequence of discrete lexical units (and structure signalling devices, such as prosody, of course)

(2.1) *The classical speech recognition formulation*

```
pressure wave ---> string of lexical items
      phoneme ---> lexical item
  lexical items ---> sentence
```

Pragmatics are important to any study of *a* discourse, but we can and must separate it out from semantics and syntax. That was suggested by Chomsky in his *Aspects* theory (1965) which defined out a complex symbol (a first approximation to a terminal symbol in sentence generation) as a matrix of semantic features, sensitive to the frame in which it appears; sensitive, that is, to syntactic features,

(2.2) *A selectional rule (complex symbol subcategorisation)*

```
                     [+Abstract] Aux-
[+V] --->     CS/(  [-Abstract] Aux-   )
                     -Det [+Animate]
                     -Det [-Animate]
```

Here we see in operation a mechanism for specifying the semantically-based selection of a syntactic category *verb* on semantic features, [+animate]. The difference between this method of constraining the generation of sentences (or, parsing those sentences, to go the other way - perhaps) and using pragmatics is in generality. (2.2) does not specify *a* verb, to be used alongside *a* noun, but rather extends over the possibly infinite set of sentences at that stage of generation. Pragmatics applies rules to objects (occurring within a universe of discourse), and the corresponding rule-governed behaviour is not described in terms of sentential structure, but as the content of propositional form. A typical question here is if *that* chessman may move to *that* square, bringing into the recognition process knowledge about a rather particular state of affairs in the world. This is a shift from specifying the form of language to deciding on what content we may 'fill out' a sentence with - here constraining deictic function.

Either syntax or pragmatics can constrain, or help choose from competing hypotheses at the sentence level. The choice here, if performance is at a priority, depends strongly on the choice of the universe of discourse; it would be perfectly valid for a suitably simple universe, we could make use of a particular constraint in there being, for example, no range of *noun* under the category of [+animate] apart from one (the user of the retrieval system) - taking document retrieval systems as an example. So the selection rule (2.2) would become obligatory in application, and in that case would usefully be collapsed into the pragmatic rules. Similarly for many rules that would otherwise be considered to be insights into linguistic competence, significant generalisations are inefficient if no generalisation is required. If we look at the effect on the pragmatic component, for the same example, there would be a transformation

(2.3)

from

ASKS(*<user>*, *<subject>*, *<author>*, *<from>*, *<to>*)

to

USERASKS(*<subject>*, *<author>*, *<from>*, *<to>*)

for a discourse universe where the class of [+animate] nouns has only one member: *<user>*. This presents us with no significant generalisation about language, the world, the semantic of questions, or even about the nature of document retrieval. A change like this, motivated solely by efficiency considerations, cannot be expected to yield anything interesting about these things. From a different point of view, artificial intelligence and computational semantics, there is no motivation to discover the structure of language, we find no valid reason for keeping pragmatics out of syntax. The attitude is very much that a system only is judged on its actions (like the ARPA projects), leaving a confusion over the rightful place of the various rules and levels of rules. It is not that the resultant systems cannot be made to work, but rather that few generalisations result. For example, Wilks (1976 p117) discusses whether or not research in artificial intelligence and the resultant systems that appear to be about language is really about language:

> an increasing number of systems in AI being designed not essentially to do research on natural language "front end" to a system that is essentially intended to predict chemical spectra, or play snakes and ladders, or whatnot . . . It is clearly the case that any piece of knowledge whatever could be essential to the understanding of some story. The question is, does it follow that the specification, organisation and formularisation of that knowledge is the study of language, because if it is then all human enquiry from physics and history to medicine is a linguistic enterprise.

This was the problem with the 'understanding' goal of the ARPA projects, a problem that faces all computational systems at some stage. On a restricted data set (universe of discourse), we can construct an appropriate model of that data (a grammar); but one of the tests of that model as scientific theory is how it handles data other that the original set. That is the difference between explanatory power as opposed to observational accuracy. Clearly, the generalisation expressed by (2.3) is not conducive to greater explanatory power in accounting for language, although it leads to a simpler set of rules *for this data*. In the quotation, Wilks was groping towards a reidentification of one of the oldest errors ever made in the study of language, that of confusing what is said with what can be said.

Meaningful comparison between the various ARPA projects is difficult because of the different stances taken on the 'trade-off' question raised above. However, the effort was *directed* towards a syntactically-controlled, pattern-recognition approach to the problem of speech recognition and its methodology. By itself this was an interesting and welcome break from the paradigm recognition systems available until then. Previous research work focused on isolated word systems, employing small computers, or hybrid digital/analogue technology, hosting conceptual solutions that did not make use of the specific properties of speech as speech (rather than just an acoustic event).

ARPA clearly identified the need for specific linguistic constraints, and in the course of the project created a demand for a useful description of English to employ in the systems, together with the means of handling that information by computer. It was unwise to sustain four major projects, CMU Harpy, CMU Hearsay, BBN Hwim and SDC, at the same time. The need for the same knowledge was felt by all projects simultaneously, and were solved in very similar ways. A better organisation of research programs would have been to replace an intensive drive to meet goals (which were not always the best conceived as we have seen) by an succession of projects.

2.3 Review: The linguistic sciences on prosody

Post-Structuralism The traditional Structuralist view of the nature of linguistics and the aim in constructing grammars was overturned by the rise of a new American school of linguistics, the transformational generative school, heralded by the publication of Chomsky's *Syntactic Structures* (1957). The structuralists' position was based on a central notion of language element identity being solely dependent on delimitation by other elements. This lead to an essentially static representation of the language system, and fostered a strong emphasis on discovery procedure, on how to arrive at the description in the first place.

The paradigm shift brought about by the transformational generative grammarians is well-documented in almost every introductory text to linguistics, and needs no discussion here. The main results of interest are:

1. The systematic accounting of grammatical structures from general, underlying rules. Previous grammars classified observable differences in a particular corpus and built a taxonomic structure on those analysis primes. This shifted the emphasis away from describing a restricted corpus to accounting for Language (however adequately).

2. The formulation of grammar using a terminology and symbolism much more convenient for computer manipulation. Much of the force behind the rise of transformational generative linguistics derived from the power of the generative/rewrite rule used in grammar.

3. The shift from describing a limited corpus to (attempting) to account for *all* sentences in a language is necessary for the machine analysis of natural language. Previous grammars on structuralist lines could only guarantee adequacy within the data it was developed from. The transformational generative promise of being able to *computationally* derive all possible sentences, however tedious or complicated for the human, was very attractive to computer scientists. Given the power of the digital computer in being able to generate list-grammars from just such finite rules that constitute a transformational generative grammar, the idea arose that an automatic speech recognition system could have the operational equivalent of an unrestricted look-up for all possible sentences.

This switch from taxonomy to generative power has the welcome effect, as far as automatic speech recognition and knowledge sources are concerned, of providing the conceptual framework which makes the rigorous formulation of grammar easier. We now have a form of description which is amenable to automatic application and testing. Automatic testing can be made easier by setting up generative rules within a computer system and giving free rein to the deduction of all possible sentences of the language which is defined by our grammar. The synthesis of such a list-grammar provides an internal check-list for known, valid sentences. This is the basis of any linguistically-centred system of text parsing. Strategies are evolved in order to reduce the sentence-generation to a fraction of the total language: we, or a computer, can search for the legal paths through a phrase structure graph formed from the grammar - this is a specific example of parsing. Also, instead of using known sentences to aid in writing grammars, as used by the American Structuralists, we turn the problem round and use a known grammar to help in analysing unknown sentences - this is the impetus behind syntax-driven recognition. However, the main contribution of this school may lie in the methods and tools of description it brings to linguistic analysis; the problem now is in deriving the content of an intonation theory to describe in a generative grammar.

Palmer There has always been an interest in the prosody of English (or meter, rhyme or versification as it was sometimes called). The literature for English reaches back to the text of John Hart (1569), though few writers on English have had nothing to say on the subject. The British school of thought on suprasegmental phonology was codified by Palmer (1922) who tackled intonation as a system, in which the units of the system have a functional identity where this identity can be determined by a consideration of their use. As with much work in linguistics, the formulation and use of an effective representation system was a concomitant of advance in theory. (On the question of graphical representation, see Stewart (1976) for a discussion of the thesis that adequate linguistic models are a logical presupposition of adequate models of language, and, further, that some graphic representations provide better models for linguistic science.) Interest here is restricted to an interest in graphical notation in so far as it can convey a phonetician's transcription of speech. The common device of representation on this school is the linear representation - Palmer moved from interlinear to intralinear transcriptional distinguishing four main nuclear tone shapes - and three types of head which effect a secondary classification on the nucleus. A retrograde move was later taken by Palmer to remove this twin categorisation by replacing the six resultant categories by 'patterns', for pedagogic reasons. The names given to these are designed to reflect the graphic shape of pitch movement in the (now combined) head and nucleus.

Armstrong and Ward An approach which has had great influence in the area of language teaching was that of Armstrong and Ward (1926). The main points here were two-fold: the division of the English phonological clause into two types, the tunes I and II of British theory; and the subordination of sentence-stress to intonation. (These two tunes are to crop up again in many unexpected places; one interesting theoretical position is Lieberman's 'markedness' account of intonation.) However, the phonological approach that was tentatively arrived at by the categorisation of Palmer gives way here to a narrower transcription of the suprasegmental events occurring in speech at the expense of phonological analysis. Of course, this is an advantage for pedagogical purposes, where the underlying system is important only in so far as it gives rise to the 'accepted' way of speaking.

Kingdon The work of Kingdon (1958) and of O'Connor and Arnold (1961) carried on the approach of identifying the functional units of the intonation system as attempted by Palmer, but with extensive additions. The head unit was divided into pre-head and body, units which could take on realisations in the speech stream according to patterns, patterns independently considered within the phonological clause. This gave rise to the notion of considering the clause as a sequence of functional intonational units, as opposed to a simple, monolithic pitch movement on a prominent syllable, surrounded by phonetic events not entering into a system of contrasts. There is a strong parallel here between the exhaustive analysis of speech into (segmental) phonemes and the analysis attempted in this project. The choice of analysis came down to between the two tunes of Armstrong and Ward and the concept of an utterance's suprasegmental features being a realisation of a sequence of underlying elements; from a linguistic point of view, the second possibility holds far more promise. By the 1960's, O'Connor and Arnold had provided a consistent and relatively exhaustive description of the intonation system. Like similar work, this was designed to be helpful to students of the English language in demonstrating the appearance of the surface structure of language.

Crystal (1969) presents a position which is in contrast to Halliday's description of British English intonation (1961 and discussed in section 3.4) in that he adopts an almost pragmatic approach towards describing the British English intonation system. He discusses:

1. The survey of literature on the field
2. The acoustic parameters involved in intonation, tone, pitch-range, loudness, rhythmicity and tempo
3. The systemic nature of the prosodic systems themselves

Intonation is presented as a complex of features from a nexus of prosodic systems. There is an assumption about the primacy of a single contour, and due consideration is given to the range of discreteness associated with the features and their systems. The theory presented is in the British tradition, as from Palmer to Halliday, and the account of structure which is given is familiar enough (figure 2.1).

The tone unit contains one and only one nucleus, which is realised by selecting from a finite number of pitch glides. The directional type of these glides is the *tone* of an utterance. This is a process of division into immediate components, and leads to a typology of tones (enumerating a closed set of glides that selection can be made from), and the ability to handle these tones and other components as being independent of place within a tone-group. The direct consequence is that place in a tone-group can now be described as *syntax* as it is not intimately bound to the *content* of any place. That is, we are now free to use even the crudest of notions of 'syntax' - the tone-group can be looked upon as a string of empty 'slots', and the tones as 'fillers' for the slots. Crystal presents the tonal system in detail giving the realisation extents of the elements of tone group. (Table 2.1 gives an idea of the general Systemic school analysis deriving from Halliday's work and modified by Crystal.)

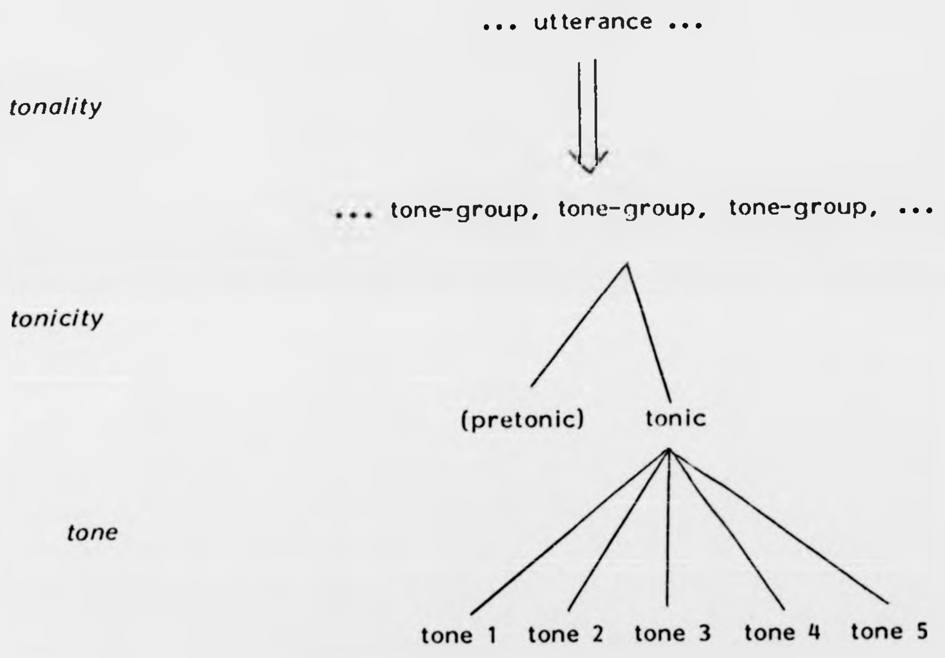
Inter-tone relations are analysed in two ways. The first analysis is part of the account given about tone which was about a typology of components occurring in a tone-group. This rests on a fact of the theory itself. For in that theory we recognise no higher unit than the tone-group, so there is no structure (of higher units) that we can say

element	*expounding syllable(s)*	*obligatory*
prehead	start --> onset	no
head	onset --> nuclear	no
nuclear	most prominent syllable	yes
tail	post-nuclear --> end	no

Table 2.1 *Utterance partitioning (Crystal 1969)*

the tone-group goes to make up. We have reached the highest level of linguistic description allowed - all we can do now is describe the

tonality	identification of tone groups in utterance
tonicity	division of tone group into tonic and pretonic
tone	identification & classification of nuclear tone



key	⇓	identified with
	^	classification
	( )	optional element

Figure 2.1 *Constituent (intonational) structure of utterance: Systemic*

occurrence of tone-groups in utterances. However, the utterance is a pre-linguistic notion and fails to exhibit any usable structure; we are forced to employ a statistical record of occurrence and of occurrence of one type of tone, given a previous tone (ditone transition frequency). Ditone transition was the most complex relation to be studied as more complex transitions (such as the frequency of occurrence of a tone, given the previous two tones - tritone transition) would involve much more work. The combinatorics of tone sequence rules demand that greatly increased sample sizes are used for a possibly small, perhaps insignificant, increase in useful information.

The second analysis of inter-tone relations is a description of tonicity, constructing a grammar of intonation out of the collocation relations and the relations *this* set of rules has with other parts of the grammar. It is one of the contentions of this thesis that a statistical analysis of language is logically dependent on a prior grammatical analysis. The question of the identity of elements of language, whether deep or surface structure, must be assumed before we engage in recording the frequency of their occurrence. To alter the definition of what we are counting, midway through statistical activity would be to nullify any results forthcoming.

Ironically, a work of this complexity and theoretical latitude is of less value to investigative research than a stark hypothesis amenable to more direct verification. Undoubtedly language is not a simple object; yet an account like this is awkward to use in active investigation simply because of its honesty in presenting delicacy in both subject and framework: Crystal discusses prosody, Halliday gives us an hypothesis. The Hallidayan position is outlined in section 3.4 and 3.5.

2.4 Prosody in automatic speech recognition

Due to a heavy theoretical and practical emphasis on the words of language, fostered by studying written text, prosody has been neglected in linguistic science, and hence in recent approaches in automatic speech recognition. One of the researchers in this field, Lea (1980b), gives a list of recent attention to prosodic analysis in qualitative or acoustic phonetic work: Lea (numerous works over two decades, cited in Lea 1980a); We can add: Ashby 1978; Cohen and t'Hart 1967, 1973; t'Hart 1979; Cooper and Sorenson 1981; Crompton 1980; Currie 1980; Denes 1959; Edward 1982; Fonagy 1978; Iles 1967; Jassem 1978; Ladd 1978; Lehiste 1970; Mattingly 1966; Ohman 1967; Pierrehumbert 1979, 1980, 1981; de Pijper 1979; Pilch 1980; O'Shaughessy 1979; Vaissiere (1974 on French; 1977 on the fine-structure of the French phrase; 1980 on language-independent prosodic features; and 1981 containing a discussion on their use in automatic speech recognition); Young and Fallside 1979. However, to date, there has been little in the way of implemented systems, with the main emphasis on gaining knowledge to use.

Vaissiere 1980 follows earlier work on intonation analysis. It presents a automatic recognition program that takes account of a 'suprasegmental representation' of a sentence. Reading 'sentence' (which is a high-level grammatical abstraction, at best a transcription into traditional orthography of the spoken utterance) to mean 'utterance', this comes to:

1. Constructing a prosodic feature vector of syllable duration and corresponding F0 value

2. Selection of the F0 values for the longest syllables in the sentence
3. Addition of this information about F0 movement to the segmental representation
4. Narrowing down of competing lexical hypothesis

It is admitted that the 'use of prosody in the KEAL system is still rather experimental' (p452) and it appears that the only approach implemented is the detection of 'important' words (full words at syntactic category boundaries) by F0 contrasts. This is essentially the Lea algorithm applied in the same way as proposed in Lea (1975), and discussed below, **Lea 1980b**, as an implementation of a weaker constraint. It is unfair to criticise Vaissiere on these points as the paper was intended to compare speech recognition computer systems as models of human speech perception. However, suprasegmental information is a very important part of speech, and psychological evidence provides much of the motivation for its inclusion in computer systems with 'perception' as their goal. Also, this paper is an up-to-date account of the KEAL system and its handling of prosody; but, as will be seen again, the major detailed descriptions are of proposals, not implementations.

Lea *et al.* 1975 This was designed to be a report of work in progress taking place at Sperry Univac and funded by the ARPA contracts. The admission towards the end: 'most of the systems have not yet been implemented, and those portions that have been implemented are open to various refinements' only underlines the shortage of alternative viable systems to discuss. It is an important paper in that this was the first extensive working out of the implications and problems of using prosodics in automatic speech recognition, albeit only on paper. The *proposal* is straightforward enough in that it suggests a fuller linguistic knowledge source to improve recognition.

Motivation for using prosodic features comes from the failure of the classical analysis of speech and the fairly recent work suggesting that some degree of syntactic analysis is necessary before attempting a phonemic level segmentation. The idea is that given a (surface) syntactic structure the listener can 'fill in' the phonemic level details, so attacking the recognition problem from the top-end, with the further possibility of applying phonological rules to constrain what is filled in.

An equivalence is assumed between the speech recognition process in person and machine. The evidence for contextual filling-in comes from experiments relying on listeners' competence and perceptual apparatus. No doubt there are active constraints involved in the fact that listeners *can* retrieve meaning from an incomplete utterance, but that in itself gives no indication of the mechanism or mechanisms that result in this effect. Lea proposes that the given account is a description of such a mechanism. The next problem is what framework for recognition should be used to derive a first approximation to recognition. The solution is to use the recognition of prosodic features to segment speech into phrase-sized chunks and to locate stressed syllables within them. (Further assumptions here are: this analysis can be done, it is reliable, and it can be used - that is, there is some correspondence between syntax and language.)

The actual prosodic features used in the preliminary analysis are energy, fundamental frequency and voicing functions. Analysis at this stage includes a prosodic structure analysis which returns phrase boundaries, rhythms and stress patterns, together with a tentative syntactic analyser which returns the most likely syntactic structures. The rest of the system, as proposed, is described in detail. The relevance to this work is in the initial analysis being performed on the basis of almost exclusively prosodic information. In a way, then, a prosodic analysis can be said to drive the recognition process.

Indirectly, two other features of interest follow. Firstly, it follows from this that the recognition process is top-down, and relies on achieving a syntactic description first. The preliminary analysis was not designed to merely segment the linear speech stream into smaller portions, but rather that the segmentation was effected into structural groupings of grammar-generated structure: to that extent the strategy was well-motivated. The second point concerns the stressed syllable location occurring within this preliminary analysis. The paper draws on work showing that stressed syllables offer themselves as 'islands of reliability'. This makes sense as a recognition strategy, given that contextual information, comprising anything from diphone transition frequencies to pragmatic constraints, can only fill gaps from known to unknown. It presents problems for parser implementation, in that the recognition must continue out from these islands in both directions, and probably from several syllabic loci at once. However interesting, this paper remains a proposal, since the system discussed did not not materialise.

Lea 1980b which takes its proper place as a review, rather than as research contribution. It is a substantial catalogue of prosodic features amenable to analysis and an account of several algorithms for such an analysis, but there is no speech recognition system presented, or even a proposal for one. Prosodic cues are discussed at several levels of language representation and of the recognition process:

1. Independent access to acoustic events, by-passing the classical analysis into words
2. Resolution of ambiguities in text and purely word-sequence analyses, providing structural bracketing for an utterance
3. Identification of juncture phenomena
4. Sentence type identification, and sentence inclusion (subordination and coordination)

Algorithms developed for the computer analysis of these features are discussed in depth; the majority come from Lea's own work. They include

1. Energy extraction
2. F0 contour extraction
3. Phrase boundary flagging (using F0)
4. Stressed syllable location
5. A proposal for slight use in the BBN Hwim system

1. *Energy extraction*. This is standard parameter extraction; a large sample frame size is used - a time window of 50ms, and a high threshold is set for the F0 tracker to avoid unreliable values and a 'jittery' trace.

2. *Boundary detection*. The algorithm works by detecting the position of dips in the F0 contour of at least 5% and similar rises following immediately after; the evidence is that such a decrease occurs at the end of major syntactic constituents. The algorithm, though, operates

on speech as a sequential structure, and can really only be considered a surface phenomenon detector. The algorithm operates as an iterative search for a fall-rise in F0 tripping the 5% level, rather than a top-down, directed search. As such it is a preprocessing algorithm, providing another prosodic feature parameter as output.

3. *Stressed syllable location*. Combination of the acoustic cues of stress, F0 prominence, vowel duration and high intensity, are claimed to give a reliable indication of stress on vowels. The algorithm uses these in the same type of iterative search as with the F0 fall-rise detection discussed above, with a running correlation between the markers deriving from the sonorant energy trace and the F0 trace, with the possibility of offset between the two.

4. *Use in automatic speech recognition*. Two approaches are discussed. The *first* is to use this prosodic information as a check on an existing analysis, as an independent, though secondary, system. An example given is the reassignment of likelihood scores to competing hypotheses; candidate words are given a stronger weight if an expected F0 fall-rise does in fact occur. The *second* is to take the prosodic analysis as being of equal importance to the segmental and lexical analysis, in effect creating a parallel recognition process with its own access to acoustic events and its own syntax description. This is a similar approach to the thesis of this work, and is supported by Lea himself in earlier work. There are several reasons why the alternative weaker approach should be tried - not all of them are persuasive. Clearly, it is interesting and valid in ASR to seek the maximum amount of additional constraints in constructing and pruning hypotheses.

However, it is possible to speculate that it is the lack of a suitable linguistic theory of intonation that inevitably relegated the prosodic component to that of a secondary system. The prevalent consensus in America is that intonation is a surface phenomena with no central links with the main systems of language. On just that point, it is significant that Lea's algorithms deal only with the linear sequence of F0 features in the speech stream.

The weaker proposal (approach one above) has been employed in the Sperry Univac system. Even here it is hard to determine the contribution made by this approach; automatic speech recognition systems are complicated structures and its is unlikely that a single component could be excised at random in order to assess its contribution. (Even talk of 'component' is at a conceptual level. Also, to construct a system without supposed advantages - in the form of a new component - is to design for second best.) So it remains an open question of what improvement derives from a component as opposed to overall design. There is clearly a need for a way of assessing this type of contribution to automatic speech recognition performance without the inelegant method of detuning existing systems. It is hoped that the system outline in this thesis can provide just that. Lea summarises the current (1980b, p201) position on prosodic analysis in speech recognition:

A totally prosodically-guided speech recognition strategy has been outlined but still awaits full implementation, and there is even some merit in testing prosodics with a prosodics-only recogniser

This thesis is such an attempt, and also an attempt to determine *how* speech recognisers should be constructed.

CHAPTER 3

INTONATION MODELLING

3.1 Introduction

This chapter discusses the linguistic knowledge on intonation that can be used to constrain ASR. Much of the effort involved in this project has been designing a suitable formalism on one hand and determining the content of the formalism on the other.

The primary function of the formalism is that it should express explicit facts and relations about British English intonation. A secondary consideration is that the notation should be amenable to the computer handling of that knowledge; this involves considerations of translatability, internal representation and analysis control mechanisms, among other items. The broader aspects of formalisation will be discussed here. The secondary considerations verge nearer to implementation details and will be dealt with in the chapters on intonation rule translation and on the SDA system performance.

A development of the systemic approach concerning intonation phenomena is used as the basis for the linguistic knowledge. The original position is described in section 3.4. The following section gives an account of the theory that is to be tested later. It has a loose basis in Halliday's account of intonation systems within the systems of language, though there have been some changes in the transition from perceptual, impressionistic statements to explicit acoustic patterns that are testable by machine. What is interesting is how far the previously unverifiable pronouncements are borne out by several (independent) experiments.

There is a discussion on the theoretical position of intonation before the content of the description is presented. Some case must be made of the separate and independent linguistic treatment of intonation. The approaches described in the review of Previous work begin from some assumption about the linguistic nature of intonation.

3.2 Formalisation

The introduction and reviews of the first two chapters introduced the notion of using linguistic knowledge to constrain the process of speech recognition. So far, only the need for such a constraint has been mentioned; nothing has been said of how such a body of knowledge about intonation can be used in this way. Accordingly, this section will deal with three areas on how intonation will be presented as part of a model of a speaker's competence. The areas are language modelling, competence and performance, and description representation.

Language modelling

The first problem is what a model of language is a model of. The birth of the transformational generative paradigm brought the problem of the formalisation of language to the fore. In one sense language can be treated as though it were a formal system from the point of view of grammar. If we rule that our investigation is into a rule-governed phenomena, then all we achieve is a description of rules, if we are successful. We can set up two poles exerting theoretical pull: on one hand we have language as a well-defined system, and on the other we postulate no system underlying the events of speech. (Well-defined in that synchronically there is nothing lost by abstracting to sets, operations between sets, and assertions about these operations. Such a system is said to be a *model* of what is under discussion.) It is held here that the problem is not in selecting which position to adopt, but in what intermediate position between the two is useful in the present project. No amount of empirical argument impinges on this selection since the choice is over how to set up the framework for empirical investigation in the first place.

In what follows, language is taken as being strongly well-defined in the sense that the rewrite rules of *any* grammar that is presented provides an exhaustive accounting of the phenomena of interest. It follows that the degree of success in recognition will be dependent on this position. All error in attempting recognition will therefore be attributed to inadequacies within a given grammar; the wider question of which particular grammar would do is left unanswered. (The only qualification to that degree of conformance to a well-defined system is in the template matching. The grammar's terminal symbols are given an operational definition in that their identity is the result of a process. Error measures at this level are taken to be some measure of 'goodness-of-fit'.)

Competence and performance

Once we have agreed on formalisation as being suitable, there remains the question of the centrality of the model, that is, whether we are describing anything more than groups of acoustic events. The competence/performance dichotomy derives in the main from the interpretation of theories of language (grammars) from disciplines other than linguistics. The locus classicus of the current controversy is the stance of the psycholinguistics of the 60's over of the implicit suggestion contained in Chomsky's *Aspects* (1965). The implication seized upon was an over-strong suggested correlation between description and use: that how language is described is how it is used

(Greene 1971 contains a good discussion of that period). Since the nature of such a distinction is the concern of those interested in the explanation of behaviour, we could have predicted that the same programme of erroneous research activity in computer science would be repeated.

A cursory survey of current Artificial Intelligence work in the field of natural language processing bears this out. An example of this is the speech understanding system of Ritchie (1977) - chosen because of its thoroughness. Here the process of understanding is seen as a conversion from one representation to another: surface form to semantic representation. Recognition is taken to be the possession of (the correct) semantic representation. It is clear that any potential problems have been side-stepped simply by avoiding the competence/performance distinction completely. There is no conceptual room for the notion of competence, and the description, theory, model of language stands or falls by its performance as a running computer program.

There are many points that may be criticised here: understanding is not the possession of a representation (whatever that means - 'representation' assumes a lot); it is doubtful if even that restricted definition of semantics could usefully find a home in computer science; the surface form of language is not wholly accounted for by an orthographic line; and so on. But the main criticism must be reserved for the claim that (human) language production and perception are totally distinct processes. showing no common source of language competence. This is the heart of the competence/performance problem. The position is set out (Ritchie 1977):

> A complete model of a language user will have to specify, among other things, how semantic structure can be converted effectively to strings of words (productions) and how strings of words can be effectively converted to semantic structure (recognition) . . . The two algorithms are distinct, since the form of input used for one is the form of output for the other (and vice-versa) . . . Chomskyan linguists usually ignore the need for effective production and recognition procedures, so in a sense the models are equally incomplete answers to both problems.

This confounds mechanism with description. People use language as a tool, in order to mean, using the shared set of conventions as a means. There is no way it makes sense to identify language with either the process of coding or decoding. It does not make sense to claim that the encoding and decoding processes have nothing in common - of course they do, the commonality is the code on which they operate.

The position taken in this work is that the features of interest are features of a shared communication code. A formalisation of the code constitutes a knowledge source on which any linguistic process can operate; the structural organisation of the SDA system reflects just that point. The rule interpreter which converts phonological rules into computer-manageable representation is separate from the recogniser that uses the internal representation. How the representation of a grammar of prosody is arrived at is of no concern to a recognition process that uses it. (In a more colourful way, there may be as many ways of learning a language as there are speakers of it.) This approach has the advantage in that any formalisation is of language as a central component of knowledge, not a confused attempt at neurophysiology. If we need to interpret our stance form the viewpoint of the behaviour sciences (computer science as well as psychology), then our description is of competence.

The production rules that are interpreted by the SDA system constitute a description of speech with reference to a proposed underlying competence. The strategy adopted here is that we can construct an interpreter (parser, understanding component) to make use of the description in an appropriate way. An advantage in freeing ourselves from only considering mechanism is that we do not need to propose a model of human behaviour. The only claim is that the SDA system performs as it does, nothing is said about how people function. We can take the performance of the analyser to reflect a hypothesised correspondence between the pattern-description and a given speech utterance. Even at this level we are only making a claim about the code used, the decoding process is taken to be a province of computer implementation. The particular implementation of the SDA analyser is heuristic two-level dynamic programming, controlled by top-down using a recursive descent procedural activation, following a tree-structured representation of the SDA intermediate representation (see chapter 6).

Considerations on description representation

The specification of prosodic description, by the SDA phonological rules, is kept separate from its internal representation and subsequent manipulation by SDA (by the SDA linker and the recognition parser). However, this still leaves us with the choice of grammar notation,or more precisely, grammar type, in which to present the prosodic rules to the interpreter. (Stewart (1976) covers the field of notation.) While still undecided over the content of the knowledge component, we can chose between a range of techniques of grammar representation amenable to computer processing. The rewrite rule is used here to give a generative description (assuming some arbitrary initial symbol). The general form of the rewrite rule production is

(3.1)

<string1> --> <string2>

where <string1> is a nonempty string of nonterminal symbols and <string2> is a string of terminal and/or nonterminal symbols. An example of use is the definition of the *Illustrative fragment of the Base Component* from Chomsky (1965, p106-7)

```
(3.2)
    (i)                    S --> NP Predicate-Phrase
   (ii)   Predicate-Phrase --> Aux VP (Place) (Time)
  (iii)                   VP -->       . . .
```

Depending on just what <string1> and <string2> in (3.1) are allowed to represent, we form a list of productions of varying power and generality. Much of the choice of type of grammar will come down to the selection of restrictions in the appearance of symbols in rules of the (3.1) type. In this way the form of production determines the generative power of the grammar. Following on, the actual choice of grammar type must be from some position in the Chomsky hierarchy of grammars formed by

1. *type 0 grammar*. The productions are of the general form (3.1). This is the most general form.
2. *type 1 grammar*. This grammar contains only productions of the form:

   <sym1> <nonterm> <sym2> --> <sym1> <sym3> <sym2>

   where a nonterminal symbol is translated out with each production, also known as a context-sensitive grammar
3. *type 2 grammar*. The productions are of the general form (3.1),

where <string1> is a single (nonterminal) symbol

4. *type 3 grammar*. The productions are only of the form:

<sym1> --> <term1>

or

<sym1> --> <term1> <sym2>

where <sym1> and <sym2> are single (nonterminal) symbols, and <term1> is a single terminal symbol.

Of course, what grammar type we select depends very much on the task expected of it. One of the goals of algebraic linguistics is to determine the weakest generative power needed to adequately handle natural language. Much of the apparent disagreement over what is the weakest relevant grammar is over what phenomena of language should be admitted as relevant. (For example, claims that English text is n-symbol lookahead can always be countered by self-embedding transformation, such that a parser detects ambiguity, especially where the completion of an embedded clause is optional - is the (optional) complement part of the nested structure, or of the host?) The traditional test of adequacy has been the phenomena of ambiguity, or rather the nonoccurrence of ambiguity. Informally, we define a sentence of a language to be ambiguous with respect to a given grammar if it can be generated by at least two distinct derivations. 'Sentence' is used in the sense of a sequence of terminal symbols. Any ambiguity must be associated with a specific grammar since families of grammars can be constructed to produce the same sentence for distinct derivations.

The formalism used by the analysis system will assume:

1. The structure of the intonation description of the breath-group can be handled by a small phrase-structure grammar.
2. The possibility of contextual embedding (parenthesis) requires a context-free description rather than a finite state grammar.
3. The operation of the analyser using the grammar description should return the overall best fitting of hypotheses without special instructions within the linguistic description. Textual parsing requires only that *a* match is found: this system is an analysis system that requires the *best* match to be returned. The mechanism by which the best (overall) match is actually determined is not influenced by the grammar description. Whether the analysis operation is full dynamic programming, heuristic search, or even exhaustive trial, is not relevant at the level of grammar writing. At the naive level, the SDA system can be used *as if* it were a simple best-first system: the user need not be aware of the underlying operations.

In addition, there are simple notational conventions in force. (The usual meanings for 'defining occurrence' and 'use' for identifiers in programming languages will be retained: the occurrence of an identifier on the left hand part of a rewrite rule is a *defining* occurrence, and the occurrence on the right hand part is the *use* of that identifier.)

1. An identifier should be defined once and only once. No definition at all results in the dependent productions using the identifier in question being ignored at analysis run-time. Multiple definitions of the same identifier result in the second and subsequent definitions being ignored at translation, but with all uses of the identifier being subsumed under the first definition. Neither of these errors are

fatal but they may produce unintended results.

2. Identifiers must be used before their defining occurrence. This follows common practice in writing phrase-structure grammars where identifiers are defined immediately after their use, as in

```
a --> b, c
b --> xxx
c --> yyy
```

   If identifiers are defined before use the error is detected at translation and is fatal.

3. The present implementation of SDA requires that embedding is made explicit. All instances of recursion should be converted to explicit choices between phrase-structures of differing complexities. Implicit recursion is ignored at translation, but will produce a fatal error at run-time.

3.3 The intonation rule base

In this section the intonation model to be tested by the SDA system is introduced. From a general consideration of non-segmental phonology we derive a description of British English intonation similar to Halliday's work of the 1960's. Several changes are made in the appearance of the intonation theory, mainly to overcome problems associated with computer formalisation.

The phonemic principle

Language is a patterning of substance, it is form imposed on substance (This discussion of realising form by substance closely follows Hjelmslev 1961, especially section 13). In the primary realisation of language, that substance is sound, the medium of temporal, linear variation in local air pressure. At this early point it becomes convenient to make a distinction between phonetics and phonology: phonetics deals with the sounds of speech with respect to their acoustic, articulatory or auditory properties; phonology deals with the sounds of speech that can serve as functional units in a language. Elements of language function through contrasts, not though there being any one-to-one or one-to-many mapping from the units of phonology to those of phonetics. The theory that linguistics functions through the mechanism of contrasts, that is, through differences of substance, was proposed by de Saussure (1959). The theory became known as Structuralism as it maintained that the most pertinent aspect of language is that it is a set, a structure, of relations of contrasts.

This step effectively dispels any philosophical objection to identifying a perceptual 'same' with continually varying events in space and time: identity is founded in a differential relationship between events, not in an event in itself. The central Structuralist thesis will be used to sustain an argument directly, in section 5.4 on template matching and in the discussion in section 3.5. The distinction, between an underlying description and a realisation of that representation in observable behaviour, has been a very fruitful one to make in modern linguistics. How we make this distinction, or where the division should be made, is another question.

Whether this division, the *emic/etic* distinction, is made or justified on operational (Bloomfield 1933), psychological (J Baudouin de Courtenay, working around 1895, as cited by Trubetzkoy 1933), intuitive (Sapir 1925), transcendental (Hjelmslev 1961), or methodological (Chomsky and Halle 1968) grounds is a matter of adherence to particular schools of linguistics. One of the strongest and most convenient forms of the distinction is the classical phonemic principle (American Structuralist) holding that Sommerstein (1977, p2):

> the differences of sound that must be discussed in a phonological description of a language are all the contrasts of that language and only those

with an Item and Arrangement superstructure of theory to classify can handle these primes (Hocket 1958).

This canonical form is part of a general theory of language that takes grammar to be a many-layered structure; phonology is the description given by the lowest layer (. . . postulated to be valid in our analysis) impinging on the phonic substance. The theoretical treatment for a treatment of segmental phonology is fairly well defined among linguists. However, there is a need to argue for a phonological analysis of prosody: otherwise prosodic effects would have no status within our theory. With no underlying representation there is no possibility of developing the explanatory power inherent in a scientific theory.

Prosody in linguistic taxonomy

Any attempt to use prosody as linguistic information makes the basic assumption that such effects in speech can be handled linguistically; that is, they can be represented by a system of linguistic signs that display contrasts, and enter into paradigmatic and syntagmatic relations. Before this can be accepted, we must consider two broad strands of counter argument concerned with the form and content of description:

1. Our general theory of language (which ever one we have) will not permit us to treat prosodic features this way. For example, the Immediate Constituent analysis, prevalent among the American Structuralists schools of the mid-20th century, starting with Bloomfield (1933), had no room in their theories for non-segmental features. (No doubt, if pressed, the theoretical justification would be something along the lines of argument 2. below.) This reduces our theory-formation to a matter of accepting doctrine, and can be countered by simple rejection.

2. Even if there was room in the theory, the particular features we are interested in cannot be represented as (linguistic) signs as they lack certain essential properties. Because of that, the features are of no intrinsic interest to a linguist, however much they interest psychologists, sociologists and others. An example of this is the position taken by P Lieberman (1967, and discussed below in this section), and Bolinger (over several years, especially 1951, 1961).

These will be taken in turn.

One of the recurring themes of modern linguistics is the 'placing' of the lowest layer of language description. We can identify two main strands in Post-Bloomfieldian theoretical positions: a marked shift from 'mentalist' and operational definitions; and a questioning of the articulatory basis of phoneme theory. Both these moves constitute a change in what is accepted as linguistic theory and each will be examined to determine their relevance to the theoretical handling of prosodic features.

1. The shift from a single operational principle or definition for identifying a linguistic entity can be effected by adopting a transformational generative grammar as our language description (Chomsky 1957). A transformational generative grammar replaces the notion of 'layers of representation' with notion of a dynamic sequence of productions, with no single intermediary production being identified as the level of phonology, or of morphology, or of semantics (Schane 1974 for an exposition and criticism). There is nothing inherent in a transformational generative grammar that makes it superior to others, on this count, at least. It is just one of many types of grammar that does not prejudice itself on the structure of language (as opposed to language-theory).

Such a shift allows an explicit handling of non-segmental information and representation, something that was not at all easy under an analysis protocol using (segmental) phonemes as theoretical primes. In that sort of analysis prosodic features were left as residue; there was no other label to apply to them (hence the derivation of 'non-segmental').

Transformational generative theory elegantly supercedes the *pitch phoneme* device which was introduced in order to salvage Item and Arrangement theories. The admittance of a pitch phoneme (Trager and Smith 1957, 41) solves the problem of residue, but degrades the the explanatory power of any subsequent immediate constituent analysis. The point of friction lies in morphophonology, as all phonemes must be accounted for in some mapping from phonemes to morphs. So to take an example from Trager and Smith (1957, 46), we consider the two sequences in (3.3)

(3.3)

/Johny/ + 2 3

and

/Johny/ + 3 2

to be different, single morphemes (following their notation for indicating the shape of the F0 contour by sequences of integers). However, any model of competence under this theory would have to assign to separate entries in the lexicon the morph JOHNY with two intonation contours, one falling and the other rising. But the common intuition is that the examples of (3.3) are of the same morph, with differing intonation (more probably, 'spoken with different intonation', or simply 'spoken differently'). A proposed solution to this objection is the device of the 'phonemic word' which has 'no connotation whatsoever of morphology' (Trager and Smith 1957, p50). In that case it cannot even begin to solve a problem of *morph*ophonology. (One wonders why 'phonemic word' was used instead of, say, 'phonemic sequence', if it was not for the sake of connotation.)

The rise of the transformational generative school (from Chomsky 1957 onward) made it easier to handle non-segmental facts in a grammar of competence. As mentioned in 1. above, the previous theoretical climate was strongly biased towards the American Structuralist school. The situation was that of Immediate Constituent analysis, with the initial analysis being made on structuralist arguments. In contrast, a generative grammar merely allows for a sequence of productions, rewritten from an initial symbol; no mention is made of levels, or of the reality of immediate productions. A grammar organised along these lines can place the same theoretical weight on both segmental and non-segmental features, and often does. One interesting development is the recent criticism (Schane 1974, throughout) on the fact that the classical phonemic level of representation is allocated no reality within the new species of grammar.

In fact, a theoretical framework like this can make segmental features logically dependent on the prior generation of prosodic features in a model of peformance. A case in point is the working of the English Nuclear Stress Rule (Chomsky and Halle 1968, p240, rule (18)). Other rules, controlling the generation of segmental features, such as vowel reduction (in the same Sound Pattern of English model) are dependent on the placing of the stress pattern. How far we can argue theoretical primacy on the basis of surface feature ontogeny within a grammar (in the process of generating its language) is moot; but at least from a methodological point of view, non-segmental features need not be treated as residue.

2. The articulatory basis of phonology is weakened by considering the non-segmental effects (phonetics) and contrasts that are realised by such effects (phonology). This amounts to no more than admitting facts to our analysis activity, facts which need to be explained once recognised. Just such a consideration is part of this thesis. That segmental effects attain a priority in analysis is not due to any of the grounds by which we set up the phonetic/phonemic distinction in the first place need not be proven. The burden of proof for that restriction should be on the move to exclusion. However, it is straightforward to present a case for inclusion.

The grounds for applying the distinction to prosody can be grouped into two main areas (with experimental evidence coming from the experiments of section 3.5):

Firstly, operational approaches in phonological analysis can be extended to include the opposition of prosodic features found in a language corpus just as much as the opposition of segmental articulatory features. So, if entry into the phonological system is by the display of contrasts and of relations with other units (paradigmatic and syntagmatic relations) in a language corpus, then we can find analogy over the classical segmental minimal pair (table 3.1 and section 6.3).

level of description	term 1	term 2
phonetic	F0 falling	F0 rising
phonology	$//_1$left for//*Italy*//	$//_2$left for//*Italy*//
semantics	statement	question
phonetic	bilabial plosive	bilabial nasal
phonology	pæn	mæn
semantics	cooking utensil	male person

Table 3.1 *Minimal pair analogy*

Secondly, we can use the same type of psychological/perceptual arguments that are perhaps the strongest in linguistics. It just cannot be said that prosodic features do not possess a perceptual identity. Once we say that they exist, and that they convey meaning, or help to convey meaning, we have all that is needed to gain the attention of linguistic analysis.

That completes the counter-arguments on the formal treatment of prosody.

Prosodic effects as realisations of Sign

We now deal with the second type of argument, on the substantial treatment of prosody. Abercrombie (1967, p89) identifies three sets of components in speech

a. segmental features
b. voice dynamic features
c. voice quality features

The segmental feature component has already been dealt with. To take the other two components - Are they, as realisations, elements of the language code? It would be a sufficient proof here to show conformance to Hocket's *Design Features of Language*. To that point we can take the feature of arbitrariness to be the most important of the design features of language (Hocket 1958; again 1960; Hocket and Altmann 1968; Lyons 1977, section 3.4). Concomitant with arbitrariness is the necessity for prosodic features to be voluntary effects. At the expense of a full discussion of the nature of the linguistic sign, it suffices to show merely that there are prosodic features, and that they are arbitrary. It is not a trivial proposition; one of the central tenets of modern linguistics is the arbitrariness of the linguistic sign. It is not the intent of this work to derive structuralism, nor to discuss semiotics, yet it is obvious that much of the acoustic raw material used in prosody could be interpreted as being involuntary events. This is something that needs clarification before proceeding. Two approaches are standard argument schemata at this point. We can present evidence on cross-language phenomena as universals of form (as in Vaissiere 1980) and invite criticism that any universal is more likely due to physiological, rather than philological constraints. Or we can show that the features of interest can be under voluntary control, at least some of the time. The second approach is used, if only for the simple reason that the search for language universals is often costly in effort and less than decisive in conclusion.

Arbitrariness is dependent on the freedom of choice in selecting the realisation of a linguistic sign. (A sneeze, for example, cannot be a candidate for being considered as he realisation of a linguistic sign. We have no control over the observable behaviour all of the time, so it cannot signal intentionality, hence, nor meaning. To take the perceived effects as an example, we can separate out levels of freedom in what is produced by the speaker:

1. *The physiological set of human speech organs*. Over the universe of acoustic events, there is a sub-set that can be produced by people. This forms an effective outer bound on speech. Within these bounds there are acoustic features that are physiologically determined and those that are voluntary. Lieberman (1967) holds an extreme position in that the physiological mechanisms involved in respiration determine the pitch contour of the breath-group almost completely. In effect, this voluntary/involuntary distinction is collapsed in on itself for intonation, leaving little choice in how F0 is produced. The contour in question that results from our physiology is a rise-fall, peaking at the nucleus. This is Tune I of the British schools (discussed in section 2.3), and the ubiquitous 'unmarked' contour of many other approaches. The 'marked' contour in this very simple system is the rising tail on the basic contour - Tune II. The reasoning behind Lieberman's analysis is the (commendably) direct relation

normal breath mechanism

vs.

increase in respiratory activity towards end of breath-group, increasing air velocity through the larynx, raising F0

which is said to signal the two language contrasts:

statement
(the normal, so-called semantically 'unmarked' speech utterance)

vs.

non-statement
(equated with the question, the only other performative use of language allowed under this analysis, see Austin (1962) for a fuller picture of language)

and

end of respiratory activity (equated with end of clause)

vs.

(signalling of) continuing of respiratory activity.

3.4 The Systemic school on intonation

One of the strongest theoretical positions on intonation was taken by Halliday (1961, 1963, 1967). This places the intonation system as exhibiting the distinguishing features of grammar: the possibility of discrete and exhaustive analysis of utterance. As such, it becomes liable to the full (Hallidayan) analysis. It is postulated that speech can be (1967 p2):

> . . . connected speech can . . . be analysed into an unbroken succession of tone-groups each of which selects from one or other of five tones . . . the selection can be regarded as discrete on both axes, both syntagmatically, and paradigmatically

(This basic assumption has been fiercely criticised by Crystal (1969b), in that 'the discreteness *must* end somewhere'. It is my opinion that this criticism results from a confusion over what a grammar is. To hold a linguistic theory which states that speech *is* a sequence of discrete units - of anything - is obviously false: this amounts to asking a description of competence to account for performance. Quite clearly, a grammar has jurisdiction over matters of system, competence and is constructed over discrete, arbitrary units. How a speaker *performs* in realising that competence is another question - a question more of phonetics, even of sociology or physiology, than linguistics. The grounds on which we can reject a grammar, or theory that leads to that grammar, are that it does not assist in accounting for behaviour, not that it cannot *describe* that behaviour. At this point of Halliday's theory the verdict can only be 'not proven'.)

Intonation is presented as a system; a label that emphasises the relational, network aspects of language. In this analysis there are four units forming a taxonomic hierarchy: each lower unit operating as an element of the structure of a 'higher' unit. Rhythm is the operation of syllables as elements of foot structure within the tone-group.

Other levels of selection concern the decomposition of continuous utterance into tone-groups (tonality); and the selection of tonic syllabic units within an utterance (tonicity). It is difficult to disentangle an explicit statement of suprasegmental phonology from this general theory of language, the systemic school of linguistics; and even more difficult to effect criticism of such phonology. Questions on how much intonation (and prosodic) phenomena can be treated as grammatical objects are more pertinent here than, say, with the construction of a structural phonotactics (as Lamb 1966). There is little point in denying that intonation patterns are not ideal language 'items'; in speech they are: less discrete, both formally and semantically; less finite (as in the opposition to LEXIS); and far more difficult to identify, whether by perceptual processes or by instrumental analysis.

However, it is misleading to base a critique of Halliday's work on such comments (Crystal 1969b, 391). Delicacy is one of the scales of analysis that interrelates the categories of theory: an increase in delicacy is a refinement in the detail of description. The introduction to Halliday's *Intonation and Grammar in British English* (1967) regards linguistics as being a position along a more/less cline, rather than a yes/no dichotomy. This, and the remarks above on competence *vs.* performance, go much of the way in meeting this criticism. The Hallidayan approach is the most serious attempt at the explicit treatment of prosody from within the language 'system of systems', and as such, is the most attractive of current linguistic theories.

3.5 Categorical perception of intonation

Halliday's statements on intonation were based on auditory impressions. However, before they are used in automatic analysis, we require experimental justification for both the hypothesised phonemic nature of intonation and for the number and stated phonetic realisation of the intonation types. This section addresses those questions for the intonation of tonic syllables. The method used here is the paradigm of categorical perception. It is found that for the experiments described in this section, categorical perception is a satisfactory way of explaining the correlation of results of discrimination and identification experiments on synthesised syllables with a variety of F0 contours. Details of the experiments and a full discussion are given in Ainsworth and Lindsay (1984). The results support the phonemic status of a small, finite set of intonation contours - in particular, Halliday's five primary tones. Also, they show strong agreement with the hypothesised phonetic forms of those tones.

Categorical perception

Experiments on the discriminability of pairs of stimuli along some continuous physical continua suggest that stimuli can only be discriminated when they can be identified as being different. This is referred to as categorical perception (Macmillan 1979; Studdert-Kennedy 1970; Liberman *et al.* 1961; Liberman *et al.* 1957). It contrasts with the classical psychophysical notion of continuous perception where the process of discrimination throughout a physical continuum is held to be independent of the process of identification (Miller 1956). For a given continuum of stimuli, continuous perception results in (possibly) many more discriminations than identifications; categorical perception results in just as many discriminations as the number of pairs that can be identified as being different.

One explanation of categorical perception calls into being a mechanism of perception that assumes identification, or classification, as basic: a subject discriminates on the basis of classification. However, that fails to explain the presence of peaks in a discrimination function at (unnatural) category boundaries before subjects are acquainted with the relevant categories (Miller *et al.* 1960; Cutting and Rosner 1974). Those results can be explained by rejecting the assumption that the auditory system must always behave linearly; regions of natural sensitivity along some stimulus continua may exist before learning (Miller *et al.* 1976; Cutting and Rosner 1974; Rosen and Howell 1981).

For natural, speech-like stimuli categorical perception has been referred to as the 'phoneme boundary effect' (Wood 1976). The initial claim was that categorical perception is speech-specific (Liberman 1957); this remains a matter of controversy, but the existence of categorical perception over a range of stimuli supports a claim of natural classes.

Synthesis of Halliday's 'Tones of English'

The synthetic stimuli used in the discrimination and identification experiments were produced by a synthesis-by-rule system (Ainsworth 1974). Just as varying two parameters during the synthesis of vowels can effectively map out a natural two-formant vowel space (Paliwal, Lindsay and Ainsworth 1982), it was found that varying two parameters is sufficient to create a cannonical intonation stimuli space for intonation containing the five primary tones. The parameters control the value of F0 for the mid (HEAD) and end (TAIL) points of the intonation contour for the carrier syllable 'yes'. The stimulus is a plausible-sounding word with F0 in the relevant region starting at 150Hz; the variable F0s ranged between 50Hz and 215Hz and the joining contour was linearly interpolated. It was expected that the F0 contour for the five tones would be

1. rising
2. falling
3. low-rising
4. fall-rise
5. rise-fall

Trial listening tests confirmed that the relevant contours were perceived as the expected auditory gesture (associated with the labels given below).

Discrimination experiments

The first stage in determining categorical perception is to measure the discriminability of physically close pairs of stimuli taken from the stimulus space. For categorical perception we expect a differing discriminability for stimuli pairs with the same acoustic separation across the physical range. An AX paradigm was used where the subject was asked to judge stimuli pairs as 'same' or 'different'. Nine subjects participated. The subjects were presented with ten repetitions of ten sets of ten pairs of stimuli covering the two parameter space described above.

The results for five of the sets of stimuli pairs are given in figure 3.1. These are for fixed TAIL parameters; two regions of high discriminability are apparent irrespective of the TAIL value. The discrimination function obtained in this way can be compared with a predicted discrimination function estimated from identification performance.

Identification experiments

Two subjects who were acquainted with intonation theory were used to provide an estimate of the identification function; they were presented with 60 stimuli ten times in random order, covering the stimulus space. The task of the experiment was to classify each stimuli as one of five semantic functions, labelled on switchboxes as

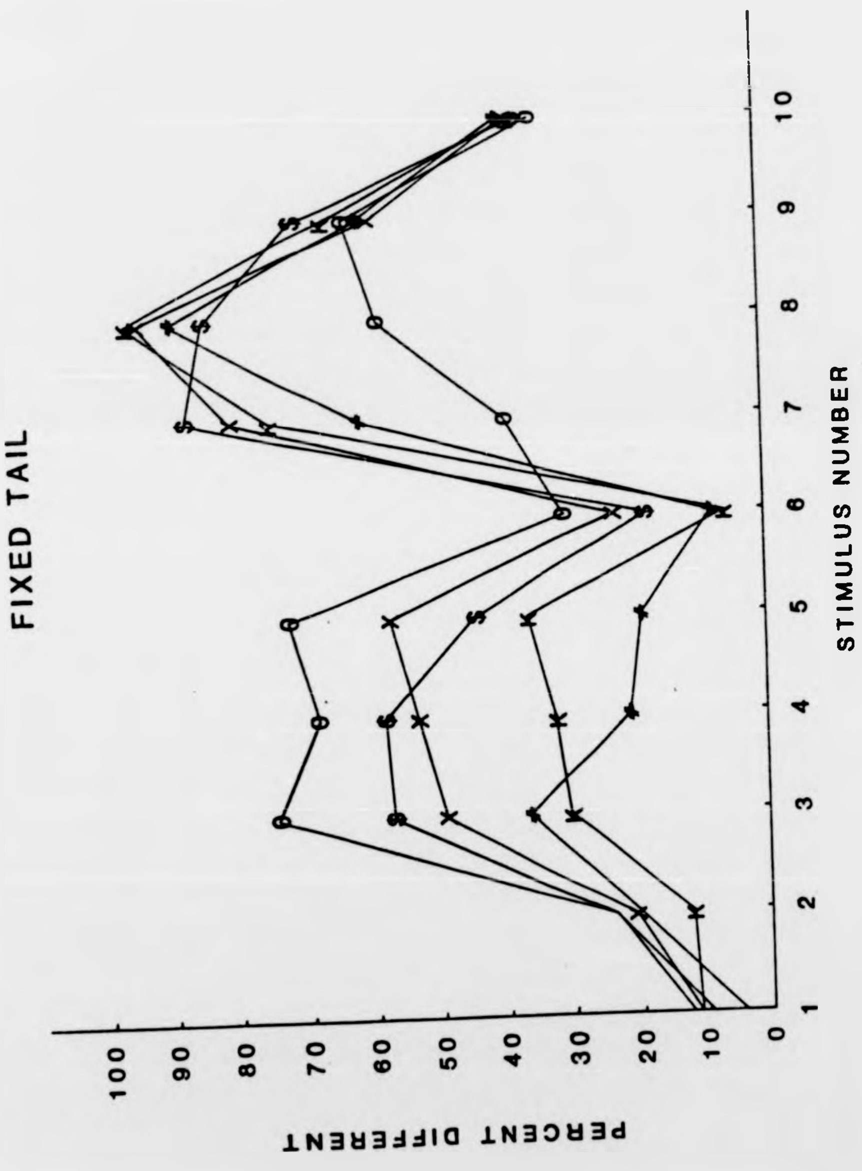


Figure 3.1 *Discriminability for sets of stimuli pairs*

1. statement
2. question
3. weak statement
4. reservation
5. emphatic statement

The results were gathered and analysed by an extension to the synthesis-by-rule system described above. They showed consistent responses with little difference between the two subjects. The switchbox responses formed well-defined areas in the two parameter HEAD-TAIL plane. When the centroids of these areas were calculated and contours constructed from these values, the resulting F0 patterns closely resemble the impressionistic descriptions given by Halliday. Figures 3.3 to 3.7 show the relevant region, the last 200ms, of these constructed contours in the format used by the computer analysis program described later.

The categorical perception of intonation

In order to show the categorical perception of intonation for tonic syllables we must demonstrate a positive correlation between the obtained discrimination performance and the discrimination function predicted from the identification experiment. The prediction formula used in calculating the predicted discrimination is discussed in Ainsworth and Lindsay (1984); it predicts discrimination on the basis of identifiability, essentially that a pair of stimuli can only be discriminated when they straddle a phonemic category. A representative example of the correlation between obtained and predicted discrimination is shown in figure 3.8 which is for fixed HEAD and varying TAIL parameter. It accords well with the criteria for categorical perception formulated by Rosen and Howell (1981)

1. a sharp categorisation function
2. a peak in the discrimination function at the category boundary
3. troughs in the categorisation function within categories

On this basis we can conclude that intonation for tonic syllables is perceived categorically, that is, it has similar phonemic properties to the more readily accepted segmental phonemes. Also, the results show that Halliday's original pronouncements on the number of primary tones and their phonetic realisation were accurate.

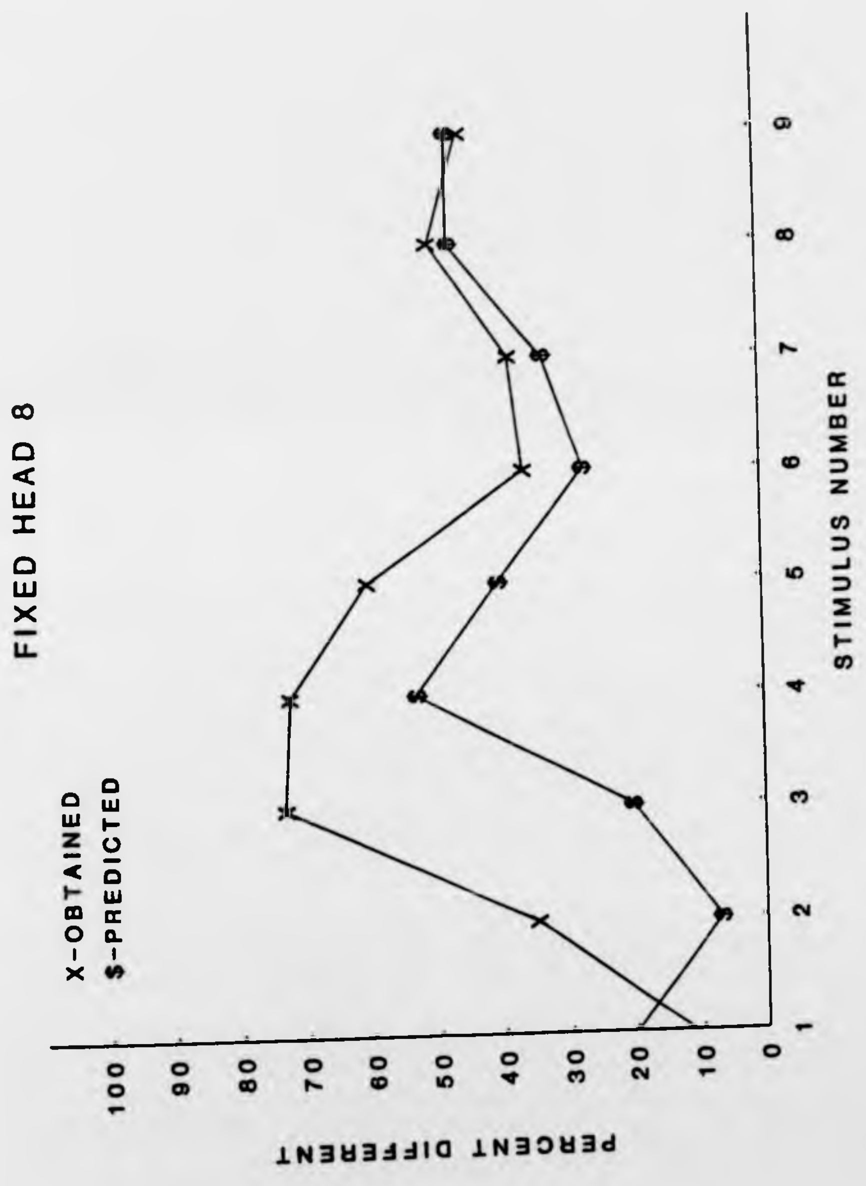
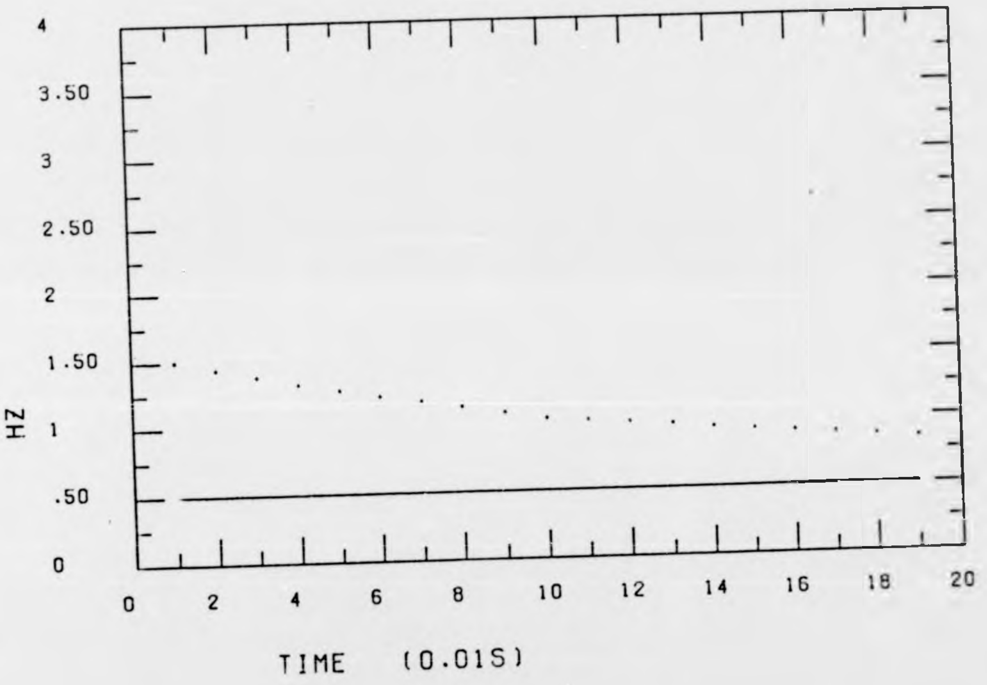


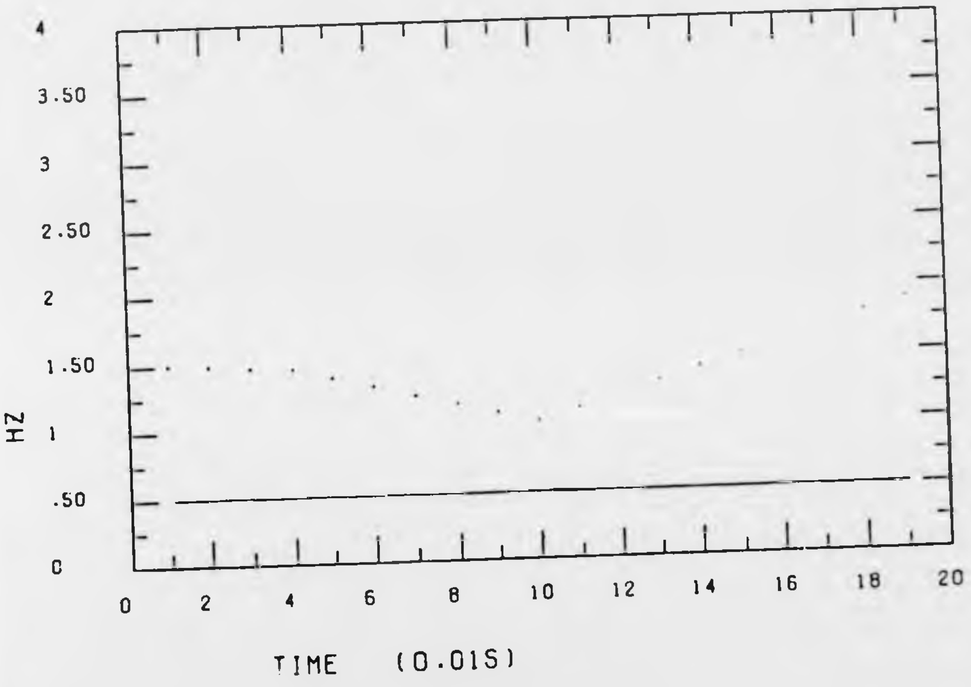
Figure 3.2 *Obtained and predicted discrimination*



XAXIS:SCALE AS PRINTED.

YAXIS:SCALE = Y * (10 **-2)

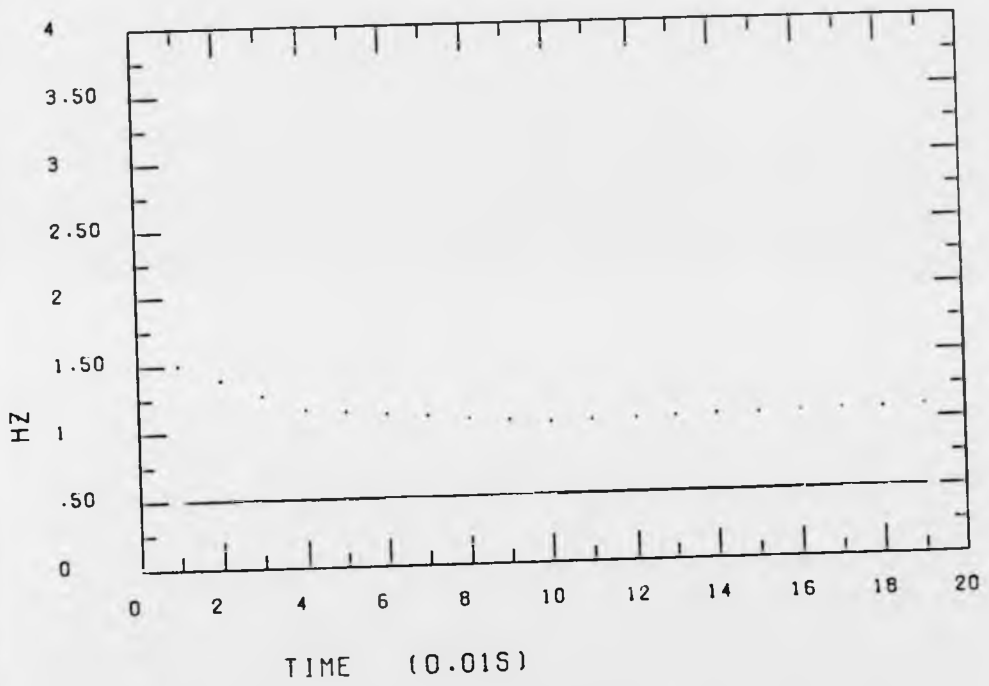
Figure 3.3 *Constructed contour: tone 1*



XAXIS:SCALE AS PRINTED.

YAXIS:SCALE = Y * (10 **-2)

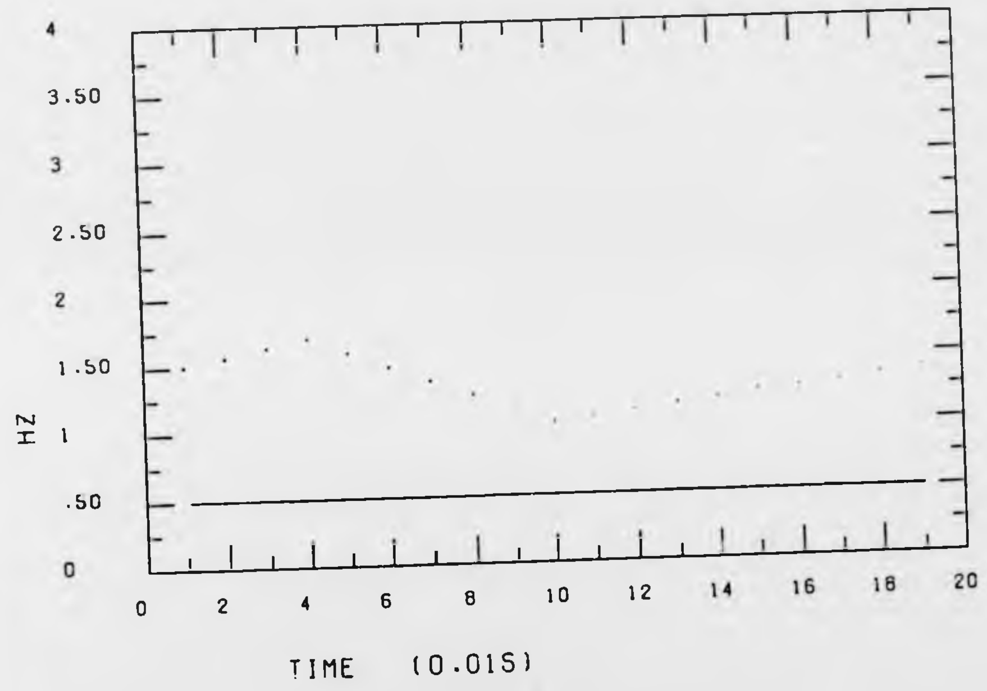
Figure 3.4 *Constructed contour: tone 2*



XAXIS:SCALE AS PRINTED.

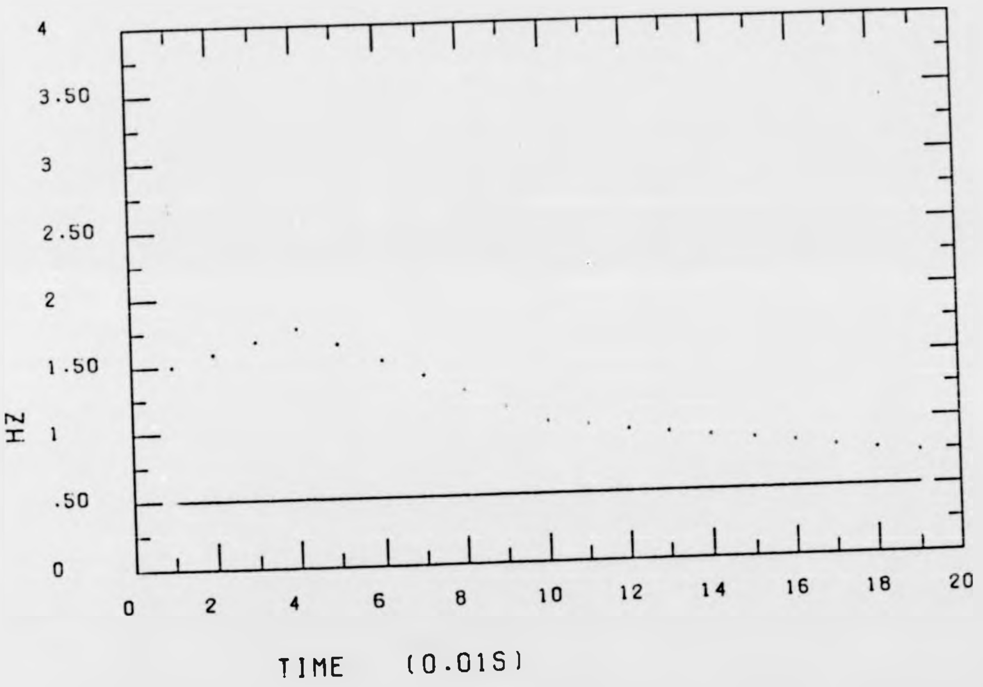
YAXIS:SCALE = Y * (10 **-2)

Figure 3.5 *Constructed contour: tone 3*



XAXIS:SCALE AS PRINTED.
YAXIS:SCALE = Y * (10 **-2)

Figure 3.6 *Constructed contour: tone 4*



XAXIS:SCALE AS PRINTED.
YAXIS:SCALE = Y * (10 **-2)

Figure 3.7 *Constructed contour: tone 5*

3.6 Intonation models for automatic analysis

The discussion on the categorical perception of intonation agrees with traditional linguistic theory; also, we have established the use of nuclear tone types as phonemic reference templates, both in the number and phonetic form expected. For the only obligatory element of the tone group, the nucleus, there is a five-way classification which is the basis of any intonation analysis, automatic or otherwise. This excludes the two remaining *compound* primary tones of Halliday's schema, the 13 and the 53, which are merely collocations of two *simple* primary tones with, however, the status of being included in the first analysis stage. Also excluded is the next analysis stage concerning secondary tones, this being a more delicate analysis which is concerned with the realisations of the primary tones. This should be handled by the transformational component of the acoustic template matcher. This five tone model is directly applicable to automatic tonic syllable analysis, where the test utterance is given to be a basic syllable and the analysis is reduced to classification. The details of such an experiment are given in section 6.3.

CHAPTER 4

THE SDA SYSTEM

IMPLEMENTATION AND INTONATION RULE TRANSLATOR

4.1 Implementation language and machine

The computer programs comprising SDA are implementations of the abstract system discussed in this thesis. The executable programs act both as a precise definitions of the system and as a working realisation. Historically, the development of both system and implementation ran parallel throughout the life of the project. Two design features in particular underwent this cross-development: these were the design of the grammar language (the metagrammar) and the implementation of the chosen analysis strategy. It was decided to design and implement a new system from the outset rather than use some existing system, perhaps not entirely appropriate or adequate. In fact, there is little in the way of comparable computer systems for linguistic rule testing *together* with speech recognition at the acoustic level. Many discrete rule systems exist for the manipulation of textual units at the lexical or morphological level; however, that results in the computer system merely performing at a symbol manipulation level. Only the large ARPA systems achieved what is aimed for here; and it was commented at the time that they were notably deficient in the ease with which the substance of the knowledge sources could be altered (Lea 1980). For that reason it is of interest to give details of this particular implementation used for this discussion before describing the detail of its function.

The high-level programming language Pascal was chosen for the programming. This choice was made on several grounds:

1. The language had to be able to handle character and string manipulation
2. It was required to possess powerful enough logical constructs to

reflect complicated control structures

3. It had to permit recursive procedure activation (more precisely, it had to be a stack-based language)
4. There was the need to have an easily portable system
5. There was the problem of general-purpose pattern recognition languages taking too much time and space on the machines available.

ISO (level 0) standard Pascal (BSI 6219) was used throughout, except for two features:

1. External file handling for the random access storage for the intermediate access is controlled in a way that is operating system dependent, as with most Pascal implementations

2. ISO level 1 (dynamic arrays) was used for some array operations: character strings and operations on arrays of reals

There are general purpose artificial intelligence languages, notably ProLog and Lisp , that may have been candidates for the present task. ProLog in particular can handle rewrite rules to some extent. The disadvantage is in the general nature of the language where all the mechanisms are designed to deal with a varied selection of rules and objects; and the 'parsing' or searching occurs in a linear fashion through the series of relational ramifications. Of course, these are hidden from the user of , say, ProLog, but they can be severely optimised by designing a special purpose system to deal with only one data type and one search algorithm, as on the case of the design of the SDA system for studying the effects of constraining the process of template matching. Also, for the present work it is necessary to deal with non-discrete pattern matching, and to some extent to be able to explicitly control any back-tracking necessary.

The SDA system can be viewed as a special-purpose language which is itself hosted in Pascal. The SDA system compares advantageously with the languages mentioned in that the SDA system exists as a collection of Pascal routines designed specifically for the task.

4.2 SDA implementation details

The Pascal used is OMSI Pascal-2 (optimising) and is run by a DEC LSI-11/23 with 256K bytes of memory using the DEC operating system RT11-XM. The language standard was adhered to since there was little need to extend the existing constructs; this policy helped in producing a relatively portable program. The suite of SDA routines consist of approximately 3000 lines of source code and compile, after optimisation, into a total of about a 200K bytes memory requirement for the programs.

The decision to use Pascal is largely vindicated by a comparison with other related systems. Friedman's (1976) system to generate language strings from an *Aspects*-type grammar compiled from about 10000 lines of Fortran to 300K bytes of memory requirement. It synthesises only, and deals with only discrete units. The Harpy system needed 10 hours of running time on a powerful computer (IBM 360) to construct its network from the language rules presented to it. The current SDA system can compile and link rules at rate of about one per second.

The SDA system programs are both suites of routines which comprise the rule interpreter and the parser, together with data representations of the network NODE and the prosodic feature vector string PFVS. The rigorous definition of the SDA system is given by the Pascal code given in appendix A4 for the rule translator and appendix A5 for the analyser. For this source code there are many lexical references to various support routines. Their precise operation is unimportant and so only their function is given here. The main routines are:

grammar translator

routine	*action*
procedure Sri	translates grammar
procedure Getnextchar	work through buffer
procedure Skip	buffer analysis
procedure SkipUntil	buffer analysis
procedure GetStr	analyse buffer into identifier
procedure GetReal	analyse buffer into real number
procedure GetInt	analyse buffer into integer
procedure Rule	analyse buffer into syntax rule
function Symeq	check next symbol in buffer
procedure Sym	get next symbol in buffer
procedure Term	analyse buffer into terminal symbol
procedure Seq	analyse buffer into non-terminal rule
procedure Rh	analyse buffer into rule part
function Cs	analyse buffer into context sensitive part
procedure Lh	analyse buffer into rule part
procedure Link	create network

analyser

routine	*action*
procedure Dp	template transformation & matching
procedure Tsh	terminal symbol analysis
procedure Analysis	language analysis routine
procedure Parse	parser

These make up the core of SDA and they share a common set of Pascal routines that initialise and access the various data constructs involved, effectively forming an environment for the core routines proper. The support routines are shown here displaying their static nesting within the program, they are not of great interest in themselves and given here only in name, which should be explanatory:

string handling

Len	Clear	Makestring
Concatenate	CatChar	Cat10
Search	Readstring	Writestring
Substring	Delete	Insert
WrString	SkipChars	Insert
CatChar	WrString	SkipChars

network data area routines

StoreA	SetA	GetA
PutA	GetA	PutA
JunkA	OkA	NewA
StoreA	SetA	BArea0
AArea0	JunkC	OkC
NewC	StoreC	CArea0
GetC	PutC	SetC
WrCNode	Wrstatus	WrSort
WrANode	WrBArea	WrTree
JunkA		

general utility routines

Stop	Pause	Alfa0
UpbAlfa	EquAlfa	WrAlfa

PFV segment routines

Gp	IMax	IMin
ConstSeg	AvrgSeg	WrTrans
RdSeg	Wrseg	Rdpfvs
Row0	UpbRow	

In turn, these routines use the Pascal run-time system for input and output services; for secondary storage; for stack handling; diagnostics for system development; and error handling. The Pascal system is operated under the DEC RT-11 XM operating system.

Dynamic Structure of SDA The procedure activation structure is given by the following procedural cross-reference:

SDA		main program		
	Calls	SetA	SetC	Readstring
		SkipChars	Substring	Len
		Writestring	Sri	Analysis
Analysis		natural language parser		
	Calls	Parse		
	Called by	SDA		
Cs		analyse context-sensitive condition of rule		
	Calls	Skip	Symeq	NewA
		AArea0	GetStr	PutA
	Called by	Lh		
Lh		analyse left hand of rule		
	Calls	GetStr	Cs	ListHead
		WrAlfa		
	Called by	Rule		
Link		create network from output of rule		
	Calls	OkA	GetA	EquAlfa
		NewC	PutC	JunkA
		UpbRow	Link	PutA
	Called by	Link	Sri	
Sri		grammar translation		
	Calls	Alfa0	BArea0	CArea0
		Makebuf	GetStr	WrAlfa
		Rule	NewA	AArea0
		NewC	PutC	PutA
		Link		
	Called by	SDA		
Rh		analyse right hand of rule		
	Calls	Skip	Seq	Term
	Called by	Rule		
Rule		analyse rule		
	Calls	AArea0	BArea0	CArea0
		Lh	Sym	Rh
	Called By	Sri		
Seq		recursively analyse non-terminal rule		
	Calls	Skip	Term	Getnextchar
		Seq	UpbRow	NewA
		AArea0	GetStr	PutA
		WrAlfa	Stop	
	Called by	Seq	Rh	
Sym		analyse next symbol in buffered input		
	Calls	Skip	Symeq	Stop
	Called by	Term	Rule	

Symeq	checks next symbol in buffered input		
Calls	Getnextchar		
Called by	Sym	cs	
Term	analyse terminal symbol part of rule		
Calls	Getnextchar	Skip	GetInt
	Sym	GetReal	SkipUntil
	ListHead		
Called by	Seq	Rh	
Tsh	terminal symbol handler in parse		
Calls	Gp	Dp	
Called by	Parse		
Dp	template matcher		
Calls	Wrseg		
Called by	Tsh		
WrTree	displays network		
Calls	OkA	GetA	WrANode
	WrBArea	WrCNode	UpbRow
	WrTree		
Called by	WrTree		

That ends any reference to the low level routines for the remainder of this work.

4.3 Implementation of the dynamic programming method DTW at template level

The dynamic programming (DP) method used in the template level matching is essentially that given by Sakoe and Chiba (1978). DP is discussed in chapter 5; only the implementation is described here. The symmetric form of the algorithm is used with no slope constraint on the time registration path (that is, $p = 0$). A DP matrix g of the intermediate optima is calculated using the DP equation

$$g(i,j) = \min \begin{array}{l} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{array}$$

which defines the interrelation of cost and state transition in the classical DP method. The initial condition, $g(1,1)$ is given by

$$g(1,1) = 2d(1,1)$$

where

- d is the PFVS distance measure, given in section 4.4 below. Its use corresponds to the log likelihood measure frame-to-frame distance used by Sakoe and Chiba
- i is the index into the speech data, ranging from 1 to I
- j is the index into the intonation pattern template, ranging from 1 to J.

The optimal score is given by the value of the endpoint in the DP matrix, $g(I,J)$. If only the value of that score is required then the amount of storage needed by the computer routine is approximately that of J real numbers; however, as we are interested in the time registration path chosen in arriving at that optimal score, we need to retain all the relevant intermediate optima to retrace the path from $g(I,J)$ to $g(1,1)$, typically J^2 real numbers.

The size of the DP matrix g is determined by the two dimensions I and J; a typical value for I is in the neighbourhood of J, giving an approximately square matrix. The maximum size of the template length can be simply altered at the program source code declaration level. The current size is 128 which gives a maximum possible template size of 1.28 seconds; this is considered adequate for handling the representation of the nuclear syllable. The corresponding DP matrix would be prohibitively large for many mini-computer systems (including the LSI-11 with 16 bit address space used here) if the full requirement of 64K bytes (128.12.4 bytes) was needed. To overcome this problem, this implementation uses one of the restrictions on the DP warping function, the adjustment window condition, to reduce the memory requirement for g without using virtual memory techniques.

Sakoe and Chiba give the adjustment window condition as a constraint on the total excursions of the time registration path by the inequality

$$|i-j| < r$$

where r is the adjustment window length. This constraint has the function of eliminating unrealistic time registration paths and the values chosen are discussed in section 6.2. The only region of g that need exist for the purpose of calculating the DP matrix is the adjustment window region and its immediate surrounds. The matching routine maps this area into a matrix the length of J, but of width $2r$. This technique results in a significant reduction in the memory required. Using the mapping function f, the standard DP equation given above appears in the Pascal code as

```
for y := 1 to ref.leng do
  begin
  for x := 1 to nsp do
    begin
    dist := d(x,y);
    if (x=1) and (y=1)
    then g[ f(x,y), y ] := dist
    else
      begin
      if (f(x,y) >= -r) and (f(x,y) <= r)
      then
        begin
        z[0] := g[ f(x  , y-1), y-1 ] + dist;
        z[1] := g[ f(x-1, y-1), y-1 ] + 2 * dist;
        z[2] := g[ f(x-1, y  ), y   ] + dist;
        g[ f(x,y), y ] := min(z);
        end (*if,within adjustment window*)
      end (*if,not initial calculation*);
    if (x=nsp) and (y=ref.leng) then tnd := g[ f(x,y), y ]
    end (*for,x*);
  end (*for,y*);
tnd := tnd/(nsp+ref.leng);
```

Further improvement in both performance and efficiency is obtained by using the Paliwal modification to Sakoe and Chiba's original algorithm (Paliwal *et al*. 1982). The original algorithm is applicable only where the endpoint I and template length J conform to the inequality

$$|I-J| < r$$

This either imposes a further constraint on the *total* time dilation allowed or an unrealistically wide adjustment window on the whole of the time registration path merely to accommodate the endpoint $g(I,J)$. The modification gives the adjustment window a slope s of the line joining $g(1,1)$ to $g(I,J)$, giving the new inequality

$$|i-(j/s)| < r$$

That same inequality is used in the formula of f in mapping the region around the leading diagonal of slope s onto a linear representation; here, again, the equation is computationally simple. The adjustment window assumes a variable geometry depending on the chosen endpoint in the speech PFVS domain and uses only a simple linear transformation to check inclusion within that window. This feature will be seen to be useful when connected template matching is introduced with its requirement of variable length matching on variable length templates.

4.4 Intonation rule translation

The grammar constraining the SDA analysis is in the form of a computer text file. This file is read by the rule compiler subprogram, sequentially reading entire rules and then translating them into a machine representation. The machine representation is only dependent on the high level data structuring of Pascal, so it is not machine dependent in any way. A typical structuring of the internal representation is given below. The end of the grammar is signalled by the location of the end-of-rule symbol (or the physical end of file). There is the option of listing the grammar as it is compiled, with the rules totally reconstructed from the internal representations. On the left side of this listing there are three numerical fields that give:

1. The running total storage requirements (Kbytes) for the internal representation. Separate totals are kept for both non-terminal nodes and terminal nodes (containing the pattern representations)
2. The internal network addresses. This option provides a check on the translator and its allocation of storage.
3. The number of the rule to which the listing text refers. This is helpful where the text of a rule extends over more than one rule.

Examples of the output listing file are in appendix A3, sections 2. and 4. These files were produced by the SDA rule interpreter operating on the grammar files given in the same appendix A3, sections 1. and 3.

The form of the grammar is defined in appendix A1 using a one-level van Wijngaarden formalism (van Wijngaarden 1974), similar to a BNF description. The (meta)grammar of appendix A1 describes a LL(1) language; therefore the translation from syntax rule (character string) to internal representation (tree node) is performed easily by a LL(1) parser. The grammar was designed with such a use in mind with the result that the choice of a suitable algorithm does not present a substantial problem. As the translation process takes place before the speech analysis proper, syntax rule parser implementation is not a major consideration either. Furthermore, the complexity of an individual rule is relatively small such that its parsing poses no problem in terms of computer implementation limitations on time or memory requirement. Accordingly, a one symbol lookahead parser was implemented by a recursive descent algorithm, of the type commonly used in computer language translators (such an algorithm was also used by Friedman (1976) in another natural language project). A recursive descent algorithm is used in the main speech analyser (although in a different implementation), so it is useful to describe the basics of that type of analysis here.

Recursive descent is a natural choice for implementation here - it provides for a simple and direct coding, with the static structure of the executable instructions mirroring the dynamic flow of control. Just as the basic rule can be seen as a composite structure (cf. appendix A1) constructed from repeating sequential elements,

string	symbol	string	symbol	string		string	symbol
lh	=	rh_1	,	rh_2	...	rh_n	;

so too can the translator which accepts this as input reflect such a structural regularity. The Pascal code is organised as a group of procedures, each with the action of translating a (repeatable) element,

```
recognise string(lh string);
recognise symbol(equals symbol);
while not end of rules do
  begin
  recognise string(dependent string);
  recognise symbol(connecter symbol);
  end;
```

which results in a target internal representation of the form:

rule representation in network

identifier
analysis status flags
rule/node type
dependent node addresses
dna_1, dna_2 , ... , dna_n
(if terminal node)
pattern template
information & analysis control

The one-pass nature of the translator parser results in a relatively straight-forward means of generating this code which will be easy to be modified in future. The constituent procedures are allowed to activate each other recursively, mirroring the form of the grammar rules. Each sentential construct is translated by a procedure which then assumes the translation of that construct as a sub-goal. The one-pass nature of the whole process ensures that the associated code for the sub-goal is generated for that sub-goal before moving onto other goals.

4.5 Extension for a transformation component

An extension to the translator code handles the extension to SDA to deal with the transformational component of the SDA grammar proper. The grammar-handling capabilities of the present SDA system extend only to the phrase-structure rules and template specification which comprise only two of the three main components of the classical *Aspects*-type grammar (Chomsky 1965). Such an extension is believed to be unnecessary for the present purpose of exploring the rules of English prosody. However, to deal with the full range of natural language phenomena, at least an extension of this type would be needed. How this addition is produced at the compilation level is described immediately below; the main cost to the analysis system would lie in the dynamics of the parsing operation. This extension is currently accepted by the rule compiler but has no effect on the intermediate representation, and so has no effect on the analysis. The current SDA version, therefore, is a recogniser of context-free languages.

Unlike the basic SDA grammar rules described in appendix A1, which are essentially phrase structure rules, transformational rules describe a change in structure that is independent of the (ultimate) content of that structure. Accordingly, the specification of structure change has two parts: a description of structure before and after transformation. The formalisation used is based on the standard *Aspects* formalism. The full form of the grammar accepted by SDA with this extension is given in appendix A2. The choice of parser implementation for the rule compilation eases the task in that it makes the extension relatively simple to incorporate. On beginning to parse a rule, the translator may encounter a transformation rule with the form:

```
[ rule sequence: structural input description
   == rule sequence: structure output description ];
```

where *rule sequence* represents a sub-tree. An example of this is:

```
        [
                # structure input description
                a = b + c ;
                b = b1 + b2;
                c = c1 + c2;
           ==
                # structure output description
                a = b1 + b2 + c1 +c2;
                                                  ]
```

The following section describes the linking procedure.

4.6 Constructing the intermediate representation

The network construction procedure (LINK) is the second stage in the production of a suitable intermediate representation to drive the speech recogniser. The first was the translation of individual rules into fragmented data structures. This second stage can be described at two levels of representation: at a logical level, taking the SDA system as a machine, or at the level of the actual computer language instructions that are immediately executable. Logically, the linker builds up an acyclic directed graph representation, a tree representation, of a presented grammar from the disjoint output of the rule compiler. At a lower level of operation, the linker can be seen as altering the relocatable output code from the rule compiler into an absolute representation of the grammar network. Such a function replaces appropriate addresses to the storage medium for the existing network such that the individual data structures are incorporated into the network. A LINK routine like this is needed because the rules are not constrained in being presented in a linear fashion. (That individual rules do not together constitute a linear sequence is not surprising as they purport to describe a highly non-linear object.) The listing of Appendix A3 gives the internal storage allocation for the network representation of the rules.

CHAPTER 5

ACOUSTIC-PHONETIC LEVEL CONSIDERATIONS

5.1 Introduction: The acoustic domain of prosody

So far this work has described the analysis side of the project; this chapter will describe the relevant acoustic and acoustic phonetic aspects of the speech preprocessing for the recognition system. There are two sides to this project: the knowledge-constrained automatic speech recognition system and the more fundamental analysis of British English prosody. Both are concerned to some degree with an analysis of the speech wave; the connection results from fitting the hypothesised templates to running speech and from designing the speech processing algorithms that constitute automatic speech recognition. In the main, the interrelated system of systems that constitutes the British English prosodic system can eventually be analysed out into clusters of acoustic feature realisations (through time) such as fundamental frequency (F0) movement, segmental duration, and intensity (Crystal and Quirk 1964). That much is uncontentious. Chapter 3 discussed the grammatical relevance of intonation (over which is less unanimity): a decision was made to limit interest at this stage of the project to F0. There are two reasons for this:

1. From the survey of theoretical positions in linguistics, surveyed in section 2.3, there is a clear consensus settling on the primacy of the intonation system among others in the event of conflict in signalling prosodic features. This primacy is observed in the way in which the intonation system can override other systems. (Perceptual experiments are used in this determination - *cf.* Fry (1958) on the trade-offs in grammatical stress perception.) Also, there is a fairly direct realisation from intonation system to pitch movement to F0 movement (not necessarily bidirectional) such that a F0 trace over

time can be considered a complete realisation of intonation events.

2. As a matter of expediency, restricting template matching to dealing with only one parameter through time makes the resulting computational effort far lighter. Also it greatly simplifies the recognition program calculation to the stage at which the implementation becomes realisable in reasonable time using a minicomputer. A typical automatic speech recognition system, especially those based on dynamic time warping, uses frequency spectrum representations of speech which requires the handling of many more data items and considerably more calculations (anything from 20 to 120 times more in each case). Therefore a study of fundamental frequency is a good choice for exploring the operation of a context-free constrained, dynamic time warping recognition system.

The current implementation of the SDA system focuses attention on only F0 movement. There is provision in the present version of the SDA program to handle three more parameters in the template matching, one of which is a confidence measure to aid F0 pattern matching. The two basic parameters not discussed here are concerned with energy distribution over time and also through different frequency regions. Although they are not used they are included in the present template specification in the SDA grammars to ensure compatibility with future systems. This chapter will describe two important aspects of the SDA system:

1. The speech preprocessing algorithms and the resulting representation. The rep-esentation is common to both the

Prosodic Feature Vector String (PFVS) and the pattern template intermediate representation that results from the grammar translation.

2. The action of the template matching algorithm. A nonlinear time-warping algorithm for matching a template onto the PFVS is presented; it is similar in operation to one-level, isolated word recognition schemes in the literature. Linear, global time-dilation of the speech pattern has been singulary unsuccessful in the past: witness the improvements achieved by applying nonlinear time warping to aid standard frequency spectrum matching techniques. However, the freedom to apply *any* sequence of local time warps has been the block on using this for practical algorithms: the sequential combinatorial evaluation of the template match scores resulting from every possible sequence of local time warps would require in the order of 10^{60} matchings (for the current SDA system). Nonlinear time warping only becomes practicable with the introduction of dynamic programming from operational research. Although a nonlinear time warping approach need not involve dynamic programming, it is fruitless to discuss practical algorithms otherwise; accordingly, section 5.4 considers dynamic programming as an integral part of the matching algorithm. The place of the one-level template matching algorithm in the full two-level SDA system is discussed in chapter 7.

5.2 Prosodic feature preprocessing

The speech data is obtained by direct analogue-to-digital conversion, sampling at 5 kHz, using an antialiasing filter, and with a word size of 12 bits. The signals were:

1. the *Lx* output from a Laryngograph
2. the output of a Revox microphone recording speech taken close to the speaker in an office environment.

Both this conversion and the routines described below were executed by a Computer Automation Alpha minicomputer using a suite of subroutines written in FORTRAN and assembler. The sampled data is transformed and reduced by a series of subroutines acting independently of the main SDA program. The processed speech representation consists of a sequence of parameter sets, Prosodic Feature Vectors (PFV), which constitutes the Prosodic Feature Vector String (PFVS); each parameter set is a transformation of a 25.6 ms rectangular time window of speech; the frame rate of this window is 100 Hz (See figure 5.1 for an outline of how the PFVS is constructed.). In the present system, the PFV consists of two measures:

1. A logarithmic transform of fundamental frequency (F0)
2. A confidence estimation on F0 (CF0)

Together, of course, with information about the position of each vector

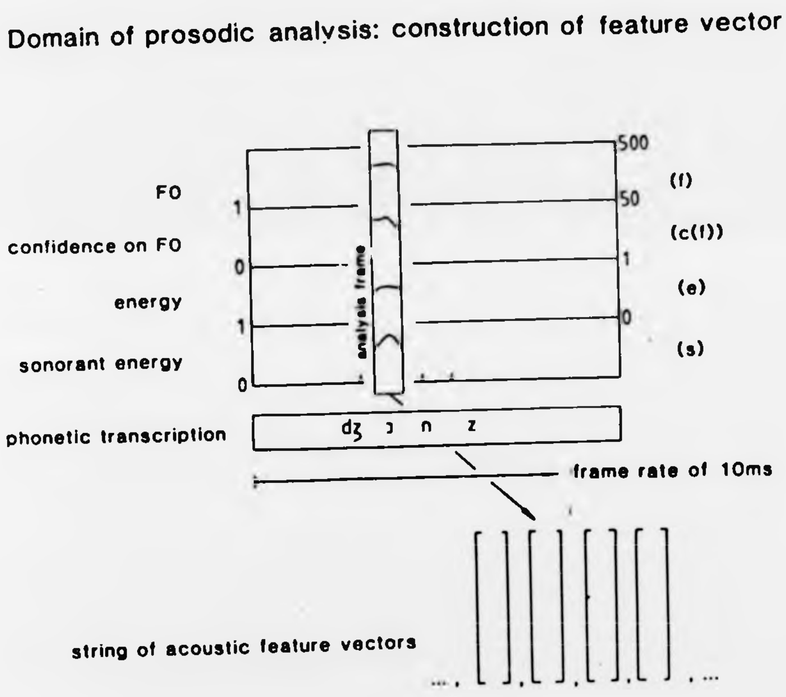


Figure 5.1 *Constructing the PFVS*

in the time-course of the utterance. Examples of the PFVS are given in figures 5.2 to 5.6 for the five Hallidayan tones of English constructed from a selection of speakers. These one second sections of speech are used in the analysis and classification experiments of chapter 6. It is apparent from these figures how good an estimator of F0 the current preprocessor is, but only in the regions where the CF0 value is above a threshold; otherwise, the forced estimate is wildly inaccurate.

1. F0 measure. The problem of determining the F0 can be conveniently thought of as the deconvolution of the glottal driving function from the speech waveform. Various means are available for doing this; Hess (1983) states that there are literally hundreds of pitch determination algorithms. Perhaps the simplest method, at least conceptually, is to forgo using the speech waveform altogether and to record laryngeal activity directly. This is the function of the laryngograph which works by measuring vocal fold impedance from electromyographic changes over time. The theory here is that the periodic closure of the vocal folds will result in a correspondingly periodic impedance variation. The technique involves physical contact between the speaker and the laryngograph in that a pair of electrodes are attached to the speakers throat in the vicinity of the larynx. The resulting *Lx* waveform through time can be analysed to give a estimate of F0 in each short-term analysis frame; the usual method here is a simple peak-picking algorithm that returns the period of the first complete peak-to-peak cycle found in the analysis frame. (The F0 estimate is very good in comparison with speech-based pitch determination algorithms. In the most thorough survey and assessment of the field to

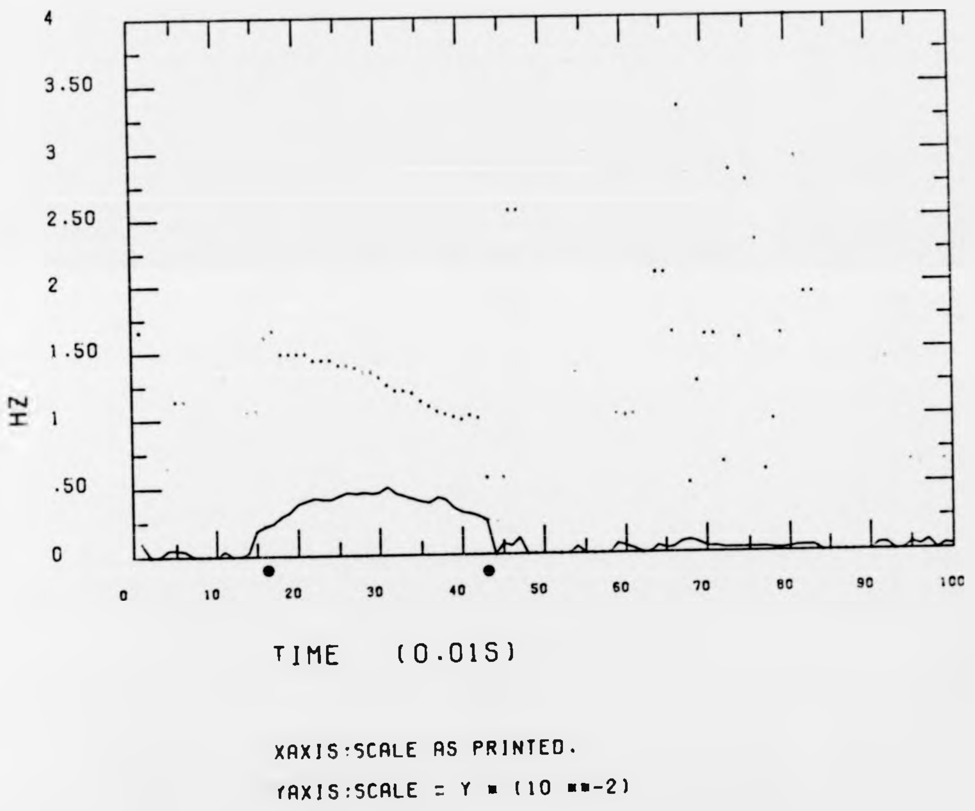
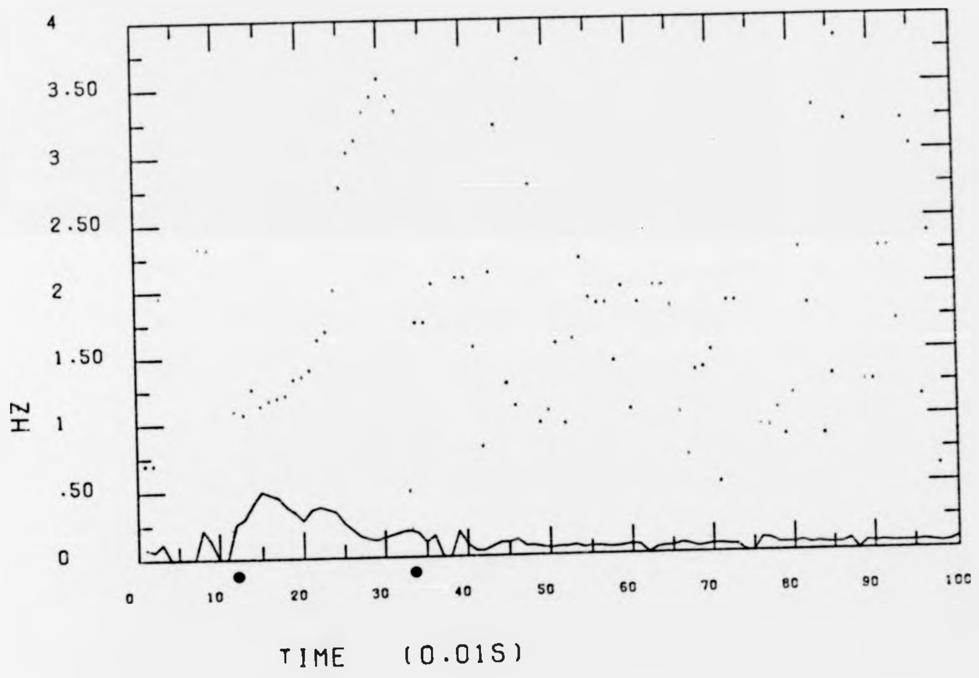


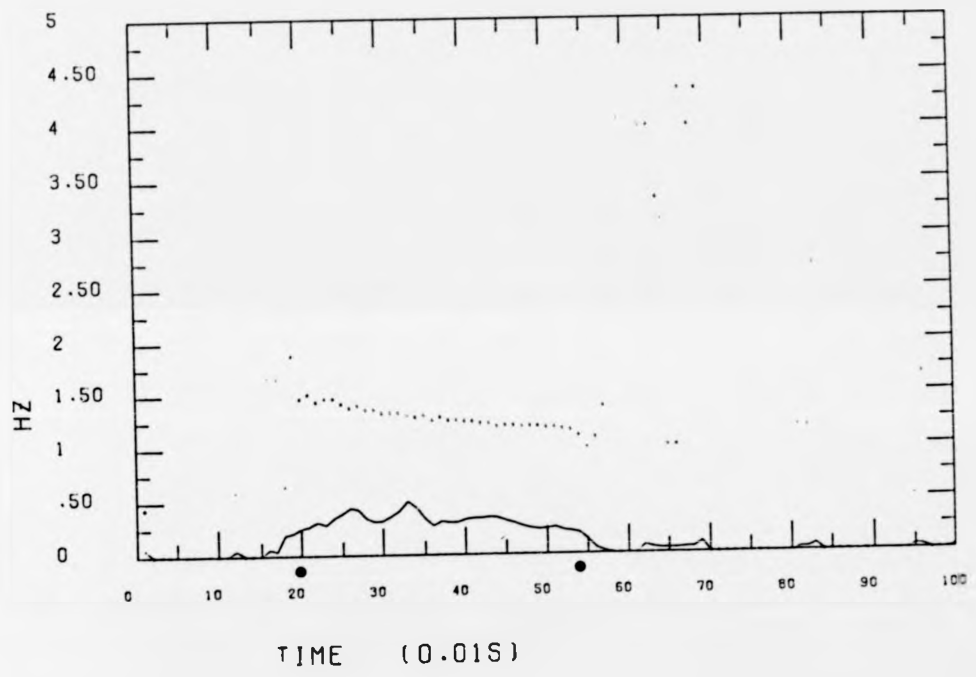
Figure 5.2 *PFVS data for tone 1. Dots indicate the F0 value estimated by the preprocessor for each centisecond. The solid line indicates the confidence measure CF0 (arbitrary scale). The region selected automatically as a true estimate is bounded by ● .*



XAXIS:SCALE AS PRINTED.

YAXIS:SCALE = Y * (10 **-2)

Figure 5.3 *PFVS data for tone 2. Dots indicate the F0 value estimated by the preprocessor for each centisecond. The solid line indicates the confidence measure CF0 (arbitrary scale). The region selected automatically as a true estimate is bounded by* ● .



XAXIS:SCALE AS PRINTED.

YAXIS:SCALE = Y * (10 **-2)

Figure 5.4 *PFVS data for tone 3. Dots indicate the F0 value estimated by the preprocessor for each centisecond. The solid line indicates the confidence measure CF0 (arbitrary scale). The region selected automatically as a true estimate is bounded by ● .*

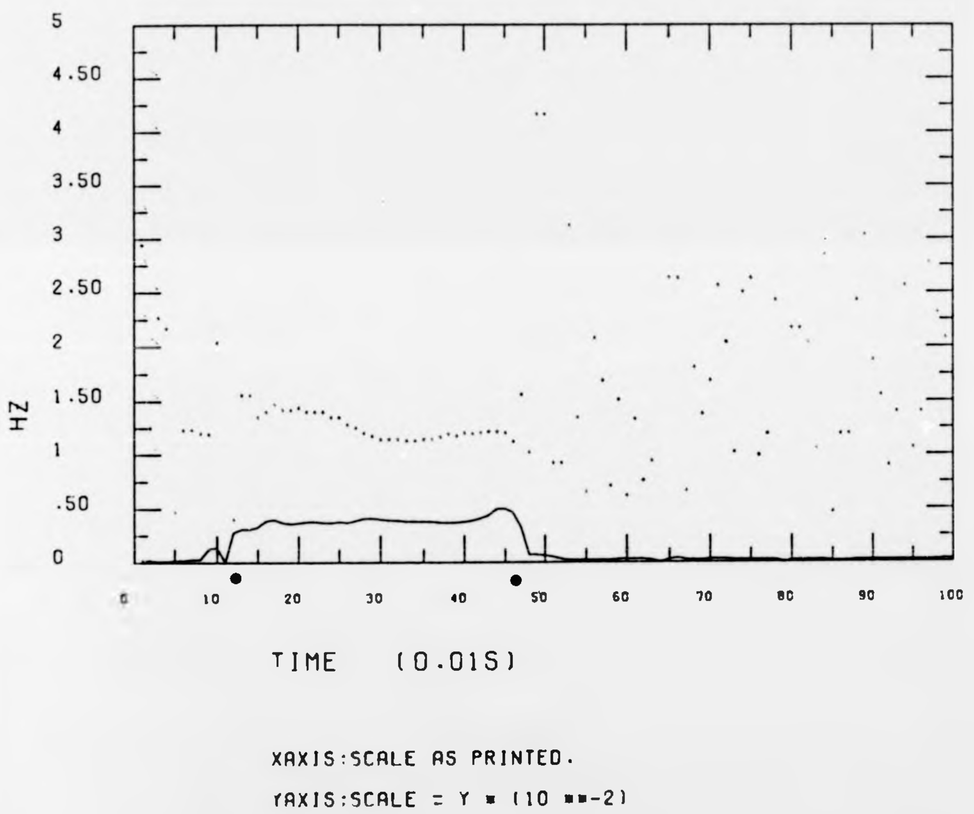
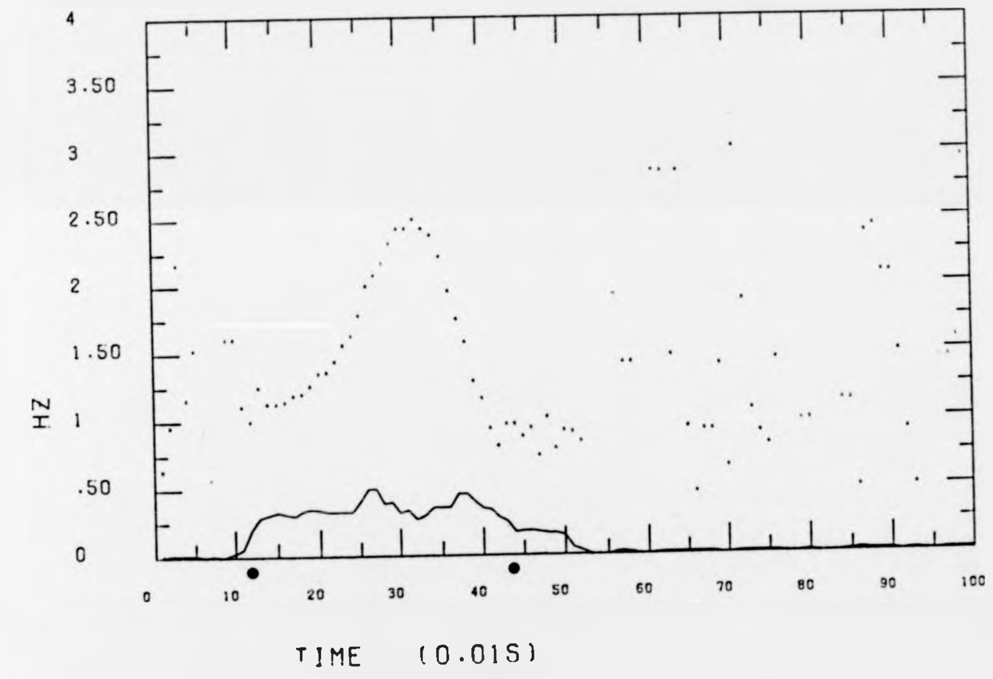


Figure 5.5 *PFVS data for tone 4. Dots indicate the F0 value estimated by the preprocessor for each centisecond. The solid line indicates the confidence measure CF0 (arbitrary scale). The region selected automatically as a true estimate is bounded by* ● .



XAXIS:SCALE AS PRINTED.

YAXIS:SCALE = Y * (10 **-2)

Figure 5.6 *PFVS data for tone 5. Dots indicate the F0 value estimated by the preprocessor for each centisecond. The solid line indicates the confidence measure CF0 (arbitrary scale). The region selected automatically as a true estimate is bounded by ● .*

date, Hess comes down in favour of *Lx*-based methods as being a standard by which to assess other methods [p508].) As the means of obtaining the feature vector is not important at the moment, the *Lx* waveform from a laryngograph is used to provide the main source of vocal fold movement representation for this project. (The instrument was built at University College, London. More details on this technique and equipment can be found in Fourcin and Abberton 1971.) The digitisation of the waveform was carried out by the Alpha minicomputer simultaneously with the speech digitisation; the apparatus allows for more than one analogue channel to be digitised simultaneously. Direct digitisation avoids the reported problem of low frequency phase distortion that the *Lx* waveform is prone to when recorded using analogue magnetic tape recording (Hunt 1978). The signal was bandpass filtered (5 - 1200 Hz). The lower figure is to eliminate undesirable slow AC fluctuations due to total larynx movement the upper figure is to eliminate spurious peaking on the waveform that may interfere with the peak-picking algorithm. (Figure 5.7 gives an example of the digitised *Lx* waveform that results in the PFVS example 5.1 above.)

(Another method is to perform the deconvolution by digital signal processing techniques as in the cepstral processing method (Noll 1964). This can operate on digitised speech without physical contact with the speaker, so is more suitable for general application. The disadvantage is that it is not totally reliable in estimation. This method was also used in the project, using a Joint Speech Research Unit computer program implementing Noll's published algorithm [Green 1972].)

2. F0 confidence measure. Conventional data-reduction as used in automatic speech recognition systems do not make full use of all information available from F0 tracking routines; they return either a frequency value or an unvoiced indication. Automatic speech recognition systems are similar in that respect in their handling of raw speech data: an example would be the common front-end to the ARPA speech understanding systems (Lea and Shoup 1980), or the voiced/unvoiced logic of the SIFT LPC routine (Markel and Gray 1976). No estimation of confidence in the F0 estimate or degree of voicing is passed on to higher levels in the system, allowing no possibility of using active mechanisms to adapt to the input sequence or of allowing the confidence in the estimate weight any subsequent template matching However, if we consider the underlying phenomena of vocal fold movement, and the observationally accessible correlate of voicing, it is clear that neither is a binary feature. In the case of using the *Lx*-based F0 estimation method the relative height of the *Lx* waveform peaks is used as the confidence estimate. The main assumption here is that vocal fold movement, relative height, is proportional to the degree of voicing; and that a high degree of voicing results in well-defined *Lx* waveforms that will be correctly analysed by a simple peak picking algorithm. The measure CF0 will be used to provide a weighting function in the frame to frame distance measure described below.

(In the case of the cepstral processing algorithm, the relative height of the largest allowable peak in the cepstrum is used as the cF0 measure. The assumption behind this equivalence is that a high degree of voicing results in a strong periodicity that will be displayed in the frequency spectrum, and so will be represented in the quefrency cepstrum.)

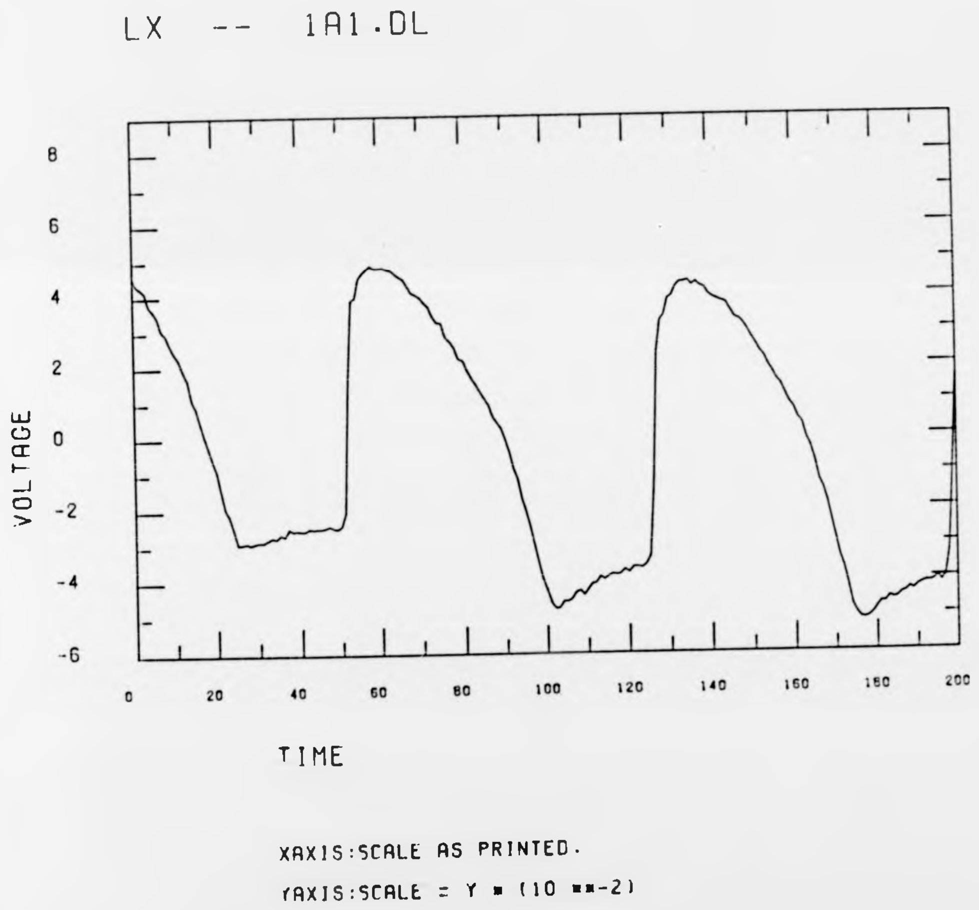


Figure 5.7 *Lx waveform for 'yes' spoken as statement*

5.3 Prosodic templates

The templates specified by a SDA grammar are, conceptually, at least, stylised segments of the preprocessed prosodic feature vector stream (PFVS) described above. That is, any matching between template and the given input looks for equivalence in as many of the features (under transformations) as possible. Goodness-of-fit is defined as being inversely proportional to the size of differences between the two PFV sections using a squared error measure. Accordingly, the templates that are given as part of a grammar must take the form of PFV representations, and must be specified that way by the SDA grammar-writer. A rigorous specification of the template part of a SDA grammar is given in Appendix A1, as that part of the grammar headed by *terminal symbol*.

```
terminal symbol: left square bracket symbol,
                 transform specification,
                 segment specifier,
                 right square bracket symbol.
```

There are two parts to the terminal symbols in a SDA grammar corresponding to the pattern template proper and the permissible operations on that template. The first part is a set of four transform specifiers. The current SDA system uses only the first, specifying the timescale match operations part of the of the dynamic time warping process (cf. section 5.5 below); the other three are described here for completeness and to show how how easily the matching algorithm could be developed further. (Future versions of the system could deal with pattern templates that specify how a matching is to be performed in terms of dynamic programming constraints on the frequency scale.) The transformations feature below in the metagrammar as four packets of upper and lower limits where each packet is of the form:

```
<no. of tries>(<min. transform>,<max. transform>)
```

For each of the four control operations to be supported by the template matcher the rule interpreter accepts the following constructs:

```
transform specifier : transform specifier;
                      transform, go on symbol.

transform : ordinal specifier,
            left bracket symbol,
            minimum extender,
            go on symbol,
            maximum extender,
            right bracket symbol.
```

The pattern template proper is referred to as a segment in the formal descriptions of this thesis. Four places for numbers in the PFV are described here, giving two more above the two required for F0 and CF0. These are positions for parameters to be used in future development and they are not used in the current system - they are included here only for completeness. The current implementation accepts a two-value prosodic feature vector as the basic pattern frame, specifying F0 and confidence on F0. The rule interpreter can accept an extended version of this basic PFV which specifies sonorant energy (referred to as SE) and mean squared energy (MSE). These have to do with energy distribution in the speech data through time - a fuller description of the extended PFV and how they are obtained is given in Lindsay (1983) and Lindsay and Ainsworth (1982).

```
segment specifier : ordinal specifier,
                    left bracket symbol,
                    prosodic feature vector string,
                    right bracket symbol.
                    prosodic feature vector,
                    go on symbol,
                    feature vector string
                    prosodic feature vector.

prosodic feature vector :
                    left bracket symbol,
                    F0 specifier,
                    go on symbol,
                    CF0 specifier,
                    go on symbol,
                    SE specifier,
                    go on symbol,
                    MSE specifier,
                    right bracket symbol.
```

The interpreter creates a variable-size area for storing the template representations, so the length does not have to be specified in the grammars. The specification of *integer* for *F0, CF0, SE,* and *MSE specifier* conforms to standard Pascal syntax for (signed) integer denotation. The acceptance routine for integer denotation for these parameters during grammar translation cannot use the Pascal system conversion routines because the translation process is heavily buffered. However, the SDA implementation code for accepting these numbers follows the Pascal system definitions given by Jensen and Wirth (1974 Appendix F).

5.4 Template matching using a dynamic time warping algorithm

One of the ways of compensating for variability in the pronunciation of linguistically same stretches of speech is by applying a complex, nonlinear time dilation to the speech (either to one or both of reference and test items). The complex, nonlinear time dilation can be decomposed into a series of local time warps; the desired sequence being the one that returns the optimal score in the subsequent cumulative frame-to-frame distance measure applied to the transformed item(s). The matching may be considered as a process of time alignment of one speech stretch to another; the optimal match score being the residual frame-to-frame matching error after optimal time alignment. As both the preprocessed speech data sequence and the templates are represented as time sequences of feature vectors it is possible to warp sequences along the time dimensions in order to minimise the summed distance calculation for the whole match. The PFVS can be expressed as a sequence of feature vectors

$$S = s_1, s_2, s_3, \ldots, s_I$$

and the intonation template can be expressed as a similar sequence

$$T = t_1, t_2, t_3, \ldots, t_J$$

A warping function can be introduced to eliminate timing differences between these two sequences; here, the warping function is a sequence of points in the pattern-PFVS time-time plane that realises a mapping function from the time axis of the pattern onto the PFVS time axis. In the case where the two items are the same, this warping function wll be displayed as a diagonal across the time-time plane, showing that minimal time distortion (none at all) is needed to achieve the optimal match. (Figure 5.8 gives an example of the time registration path that results from time warping.) The dynamic time warping problem can be reduced to discovering the optimal time warping function for a given stretch of speech and reference template such that the distance between the transformed template and the speech data is minimised. Sakoe and Chiba (1978) present two conditions for the general automatic speech recognition case:

1. Speech patterns are time-sampled with a common and constant sampling period
2. We have no *a priori* knowledge about which parts of the speech pattern contain linguistically important information

(to be more precise on the second point: even if we do have such knowledge, the system cannot use it) to which we can add, or rather, make explicit, two additional conditions for the extension to the present case of recognising speech F0 contours:

3. There is a formulation of a calculatable error function which can provide a scalar measure of distance for any data and template frame matching. Unlike the short-term frequency spectrum representation of speech and its derivatives, it is certainly not the case that a F0 value can be determined for any given stretch of speech. It is at this stage that we must use the two-level pattern information given in the PFVS. A problem with many recognition processes is in how to handle two types of information associated with a pattern: the pattern value and the binary feature of of the pattern being present at all. The usual method of handling this is by specifying the value or giving a null value which acts as an implicit indication of the feature not being present at all. This results in problems in how to measure the fit of one fundamental frequency trace against another:

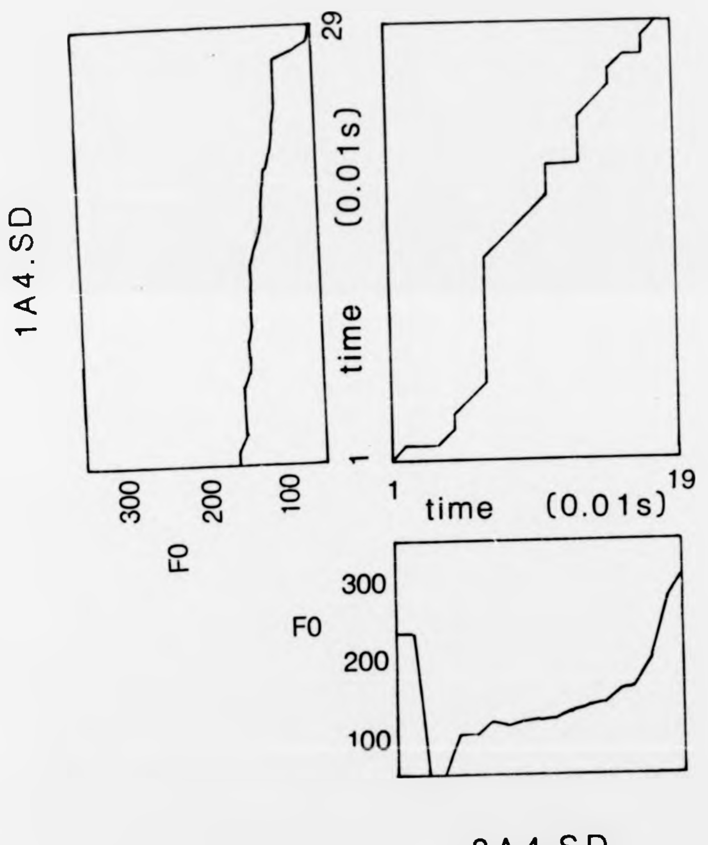


Figure 5.8 *Example of the time registration path resulting from time warping*

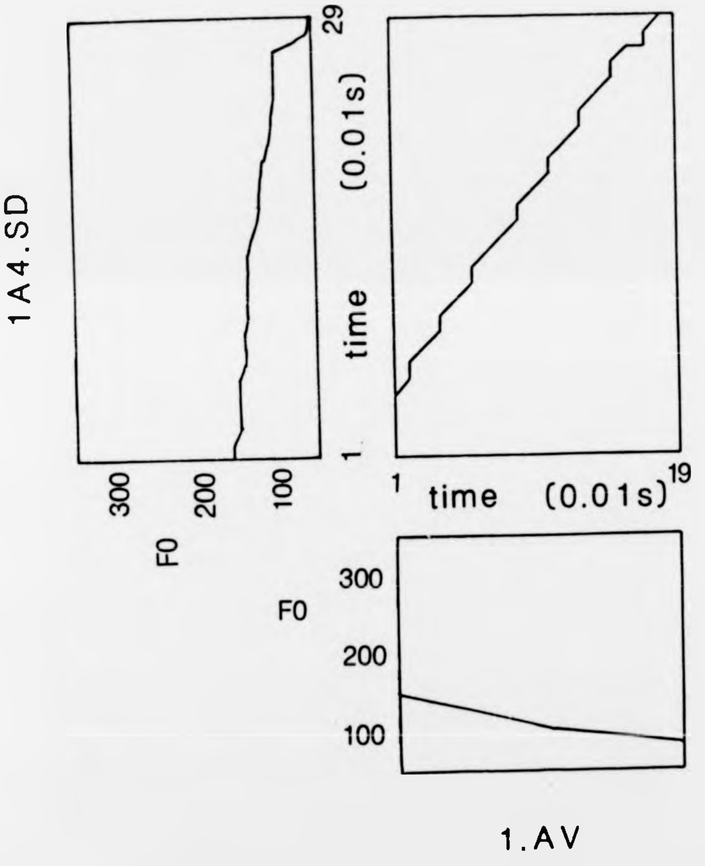


Figure 5.9 *Example of the time registration path resulting from time warping*

a mean squared error measure fails when a fundamental frequency value is compared with a value which is more properly an unvoiced indicator. A mechanism which is forced to return only a graded value is unable to handle a lack of pattern (without using pattern tracking algorithms). The distance function *d* here uses the self-estimated confidence measure *c* in the F0 value *s* to form a weighted sum of the squared differences:

$$d(i,j) = c(i) * (t(j)-s(i))^2$$

where i ranges from *lwb* to *upb* of the data segment
where j ranges from 1 to J of the template segment
c is confidence measure
p is pattern template F0
s is speech data F0

There are two main advantages to this. Unvoiced segments or even frames are handled without invoking exception handling or smoothing routines. Regions of high confidence contribute more towards the magnitude of the final recognition score, again without special handling.

4. The phonological identity of a template remains intact after the time-warp transformations. This point imposes restrictions on the nature and amount of time-warping allowable: essentially it assumes that the linguistic content of uttered language is preserved across time-warping transformations and that localised intra- and inter-speaker variability along the time scale can be ignored. It should be noticed that there are no constraints placed on the types of pattern used. In previous work the the feature vectors have been representations of the frequency spectrum of speech, derived either from a filter-bank device or from digital transformation. The

error function used would calculate the distance between two frequency spectra. In the SDA system F0 values are used as features in the matching and the distance function is a special case of the more common feature vector distance case.

It was mentioned in section 5.1 that the problem now becomes determining that optimal time alignment, the optimal sequence of local time warps. Each possible warp sequence cannot be practicably tried out in turn: a more efficient method is needed. One candidate method of determining the optimal warp sequence is to use dynamic programming (DP). This is a powerful technique imported from operational research and successfully used in ASR in work by Velichko and Zagoroyko (1970), Sakoe and Chiba (1978), White and Nealy (1976), Sakoe (1979), Bridle *et al*. (1979); and Myers and Rabiner (1981). Unlike linear programming, DP reduces the problem to that of finding the best *way* of solving the problem; that is, to finding the best sequence of state-to-state transitions through a problem state space. (Of course, once the best *way* of solving the problem is found, then solving the problem itself is trivial. In this case, once we know what sequence of local time warps give the optimal score, we just have to apply them to get that score.) The Principal of Optimality underpins DP, here formulated by Bellman (1957):

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision

where *stage* is the index of the succession of changes in the *solving* process; a *state* is one position in the solving process. The solving process state-to-state transition will be fully determined by the current state in the present stage and by the policy in force. The functional equation, for the present problem formulation, which equates the policy value for a state at a given stage, is given by

$$g(i,j) = \min \begin{cases} g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-1,j) + d(i,j) \end{cases}$$

This is a symmetric form of one of Sakoe and Chiba's DP equations with no slope constraint (that is, $p = 0$). The equation was found by them to perform better than a range of alternatives and has the advantage of being the simplest, computationally.

The SDA DP algorithm substantially follows the exposition of Sakoe and Chiba (1978). The domain of the DP equation is specified by

$1 < i < I$	for the PFVS
$1 < j < J$	for the template

and the Paliwal modification is used to give a variable geometry adjustment window to constrain the time registration path

$$|i-(j/s)| < r$$

The initial condition is given by

$$g(1,1) = 2d(1,1)$$

and the time normalised distance, the optimal score, for a match is given by

$$g(I,J)/N$$

where $N = I + J$. The DP equation is a recursive formulation. Obviously the solution score cannot be calculated directly from this equation; in practice a matrix g is formed of all the relevant intermediate optimal scores - hence the appearance of the identifier g in the DP equation. Full details of the DP matching routine implementation and the Paliwal modification on the adjustment window are given in section 4.3.

Examples of template matching

Two examples of the time registration path obtained from time warping are given in figures 5.8 and 5.9. Both examples take a natural utterance sample of tone 1 spoken as a single syllable as the reference template. These examples are taken from the nuclear tone analysis experiment described in chapter 6. The selection of PFVS from the spoken utterance is automatic, depending on threshold crossing on a smoothed CF0 contour. Any such method is only a compromise, and the smaller plots around the time registration paths of figures 5.8 and 5.9 show slight inaccuracies in F0 estimation (values given in appendix A3). Those would be lost if a manual selection system was used where the selection criteria were changed from sample to sample. However, as the example of figure 5.8 shows, these inaccuracies are sufficiently compensated for by time warping within the adjustment window and by the weighting on the dynamic programming distance function.

The reference templates 1A4.SD and 2A4.SD were found to possess the minimum intra-class distance among a selection of 32 tone 1 and tone 2 utterances, respectively, randomly taken from a database of 400 tone samples from four male speakers. This process was part of classifier training for speaker dependent and independent recognition; it conveniently provides representative samples of each tone for the present purpose of comparison. The third reference template 1.AV was synthesised from the averaging of the identification results from experiments conducted to demonstrate categorical perception in section 3.4. The F0 plot is given in figure 3.3.

1. *Tone 1 and tone 2.* (figure 5.8) This is an interesting comparison since tones 1 and 2 are held by our theory to be substantially different, and some even hold that this difference is the only formal distinction that can be made concerning intonation. The figure shows the attempt to warp one into the other. It can be seen that the time registration path has the greatest time distortion at the midpoint; this is where the two graphs have similar values. This is the attempt to match similar values around the mid point, as far as the adjustment window allows. The poor F0 estimations of the first four values have little effect on the warping function overall, possibly because they are associated with a low CF0.

2. *Tone 1 and synthetic tone 1.* (figure 5.9) This is a comparison between two representative tone 1 templates, one from recognition training and one from perception training. As expected, they are similar; apart from an initial displacement, the time registration path follows a straight line, showing that the best match is obtained with an almost linear dilation. This suggests that the two templates are similar, apart from their duration. Inspection of the values given for the two

templates in appendix A3 bears this out.

CHAPTER 6

SDA ANALYSER: DESCRIPTION AND PERFORMANCE

6.1 Syntax directed analysis

Multi-level analysis

This section describes the operation of the SDA analyser and discusses the motivation behind some of the design features. The analyser is essentially a two level algorithm consisting of the DP template matcher described above in section 5.4 and an algorithm that returns a recognition score for one or more templates selected in sequence under the direction of the syntax rules currently in force. This section also describes the syntactic constraint mechanism which operates a top-down parse of the intermediate syntax representation; it determines the *permissible* sequences of templates from which the second level of analysis determines the near-best sequence. For the simple grammar consisting of one template, or the selection of one template from many, the SDA system is effectively an isolated template recogniser - although operating on intonation contours. That operation is described in use in section 6.2 where experiments on single template training and classification are described. One example of this is the speaker dependent classification of the five Hallidayan simple tones using reference templates from training on speaker 1 (table 6.1):

```
#
sigma = a, b, c, d, e;
#
    a = [  1(1,1), 1(1,1), 1(1,1), 1(1,1),
           (

           . . .

           ) ];
#
    b = [ . . . ];
#
    . . .

    . . .
#
```

The resulting score for single template analysis such as this is optimal.

The algorithm of the context-free constrained connected recognition is, in general, unknown. The approach taken here is to apply a heuristic algorithm which returns a near optimal score. In contrast, the two level algorithm of LOGOS is a fully optimal DP solution to the problem of selecting endpoint positions along the duration of the connected template 'super-pattern' (Sakoe 1979) for the simpler finite state grammar. The current implementation of SDA trades efficiency to achieve to near optimal solution to the best sequence of templates. The analysis algorithm selects a range of endpoints for the first level template DP, for each template in any candidate sequence. This contravenes the Optimality Principle; the estimation on the locally optimal endpoint is made with little or no reference to successive choices. However, the resulting score may approach the optimal for a simple grammar of the type envisaged for intonation, and for a sufficiently broad range of endpoints at each template junction.

The main use of the SDA system in the current project is on the single template recognition experiments (section 6.2, following) which use a limited syntax of choice over a template lexicon of five reference templates. That series of experiments makes no concessions over optimality. They are good examples of how a well-motivated syntax can direct the recognition of the intonation contours. The syntax may be considered well-motivated because the linguistic classes represented by the reference templates are phonemically valid abstractions in both number of phonetic form.

Section 6.3 presents an example of SDA operating with a more complex constraining grammar. This uses a connected utterance as the PFVS and a collocation grammar as the structural hypothesis. This run of SDA stands only as an example of connected utterance use for the reasons that the algorithm only approximates optimality in this mode and that the content of a connected utterance intonation grammar has not been fully determined. It is included in this thesis as an example of how a more complex grammar, possibly describing the phonetic form of the tone group prehead, could be tested and used in future work.

Syntactic constraint mechanism

The SDA system operates with a grammar representation which restricts the choice of reference templates in the selection of alternatives, or permits only certain sequences to occur, where *sequence* and *alternative* denote either reference templates or other sequences and alternatives. The algorithm which determines the set of alternatives or set of permissible templates is separate from the two level analysis algorithm proper. The function of the constraint algorithm is to provide a set of permitted templates whenever required by the second level analysis algorithm. That set is then used in the operation of the heuristic analyser to choose the best fitting sequence of templates. The algorithm is logically separate from the analysis procedure, except where and when the permissible set is delivered.

The internal syntax representation is as an acyclic directed graph and in the current SDA system the relevant sets of templates are determined by parsing this network by a LL parsing algorithm in synchrony with the main analysis. (See discussions in Leith 1983; Fu 1977, 1974; DeMori 1977; Lewis and Stearns 1968.)

There are two ways of tackling the intermediate representation network parsing: performing the analysis either top-down or bottom-up, the direction determined by the analysis stage at which the input symbols are inspected. For top-down operation, such as LL parsing, the difficulty lies in designing a suitable grammar such that it can be manually translated into a series of mutually recursive procedure calls. On the other hand, an LR analyser needs rather complex state tables to be built; these can be constructed automatically, given a (suitable) grammar. In the absence of a program to do that work for us, the choice comes down on the side of LL analysis. The solution chosen here was determined by the lack of automatic means for generating the code for a LR(n) analyser from the combination of network and template-matching routines. The position is more complex than this, though - see next section for how the need for recognising rather than accepting terminal symbols requires a rather specialised solution. The operation of top-down syntax analysis has been described already in section 4.4 in connection with the algorithm that translates the input grammar.

The algorithm here is LL, implemented as a recursive descent routine. The choice has implications for the form of the rule syntax as described in section 3.2. Much the same concern is found in the translation of computer languages where the paradigm case is the LL(1) parse. Knuth (1971) sets out four conditions for LL(1) grammars, subsequently collapsed into one by Griffiths (1974): the most immediate of the restrictions is the loss of left recursion, typically occurring in rules of the form:

a --> a, b;

where a is a non-terminal symbol, and b is either terminal or non-terminal. Of course with this construction there is no way for the analyser, parsing from left to right, to know when to choose the symbol b - it will never get there. An analyser of the recursive descent type used in the SDA system will call itself indefinitely until the analysis stack cannot be accommodated by the particular implementation. Algorithms exist for automatically testing grammars for LL(n) properties (Bornat 1979; Griffiths 1974); and often grammars can be rewritten in order to preserve their properties of description (weak equivalence) while conforming to LL(n) conditions. An example of such a technique is the elimination of left recursion by rewriting the original production using either right-recursion or iteration (Bornat 1979, 281).

The current SDA implementation is more complex than just a recursive procedure. Recursive descent is only applicable to procedure activation within a computer program, whereas the SDA system uses a constraining grammar which is not strictly part of the computer program.The solution used here is to simulate the operation of the recursive descent parser by an interpretive top-down analysis of a graph (in this case, the node network). An interpretive top-down analysis is performed by the main analysis routine following the links between nodes in the grammar representation network. (A fuller discussion of this approach used in a restricted way for computer languages can be found in Bornat 1979 chapter 16.) This particular implementation uses n-ary nodes in the graph, rather than the binary nodes usually advocated; this saves on the duplication of control information, and with the use of variable size arrays, will be even more efficient and compact. The simulation of the top-down analyser is effected by tracing the intermediate network, following indicated dependencies between nodes. The tracing is performed by a recursive procedure which is called at each transition between nodes. (This stage is mentioned by Bornat 79, chapter 16, in passing, in connection with using a BNF description as 'programming in BNF'.) This contrasts with the usual method of explicitly stacking and unstacking the graph pointers on separately constructed data structure. By using a (recursive) call of the analyser at each follow-up of a network address, the program uses the Pascal run-time stack for retaining the current analysis information, and so uses that language's mechanisms for the stacking and unstacking. All stack management is performed by the Pascal run-time system, so making for a simpler and efficient parsing routine.

A heuristic procedure for connected recognition

In the absence of a fully optimal DP algorithm for context free structures the connected recognition algorithm uses a heuristic procedure based on the path restriction method of solving computationally impractical DP problems. The method is described by Norman (1972) and was first described by Durling (1964, cited by Norman). It resembles the hill-climbing method of many optimisation strategies and it may be relied upon to deliver a good to near optimal solution. The procedure is informally described (Norman 1972, p19):

Take a path, that is, a sequence of states, one for each stage, and in successively evaluating the functional equation at each stage, examine only states in the neighbourhood of the state on the path, improving the path if possible. If an improved path can be found, repeat the process, considering only those states in the neighbourhood of the improved path, until no further improvement can be made

The disadvantage with the method is that the algorithm may latch onto a local optimum; the corresponding sub-optimal time registration path here would consist of a series of saddle points running alongside the optimal solution. The advantage, apart from the absence of other approaches, is that a good initial estimation of path reduces the chance that the algorithm settles on a local optimum. A good estimate is achieved by choosing an appropriate syntax and using realistic reference templates.

The path restriction method operates in SDA by selecting the best score returned from the template level for a range of endpoints, continuing this procedure for all successive template collocations. The range of endpoints is chosen to lie around the original template length. The successor template begins at the endpoint corresponding to the minimum score and the process is repeated. The routine is implemented as a recursive procedure in SDA; the operation effectively forms a tree structure of single template DP matches with a range of starts and ends. Durling refers to the restricted path through successive policy estimations of this method as a 'tube'; within the tube, the scores resulting from all possible endpoints are evaluated. With a wide enough range of endpoints, this method is identified with the method of exhaustive evaluation discussed in section 5.4; the impracticality of exhaustive evaluation for anything more than a small number of policy states forced the introduction of DP in the first place.

The full specification of the process is given in the Pascal code of appendix A5. This expands on the basic notion that, for template m, of length $t_m(l)$, starting from position a of the PFVS, with a path restriction neighbourhood size of $2k$, the single template DP matching routine

$$DP(template, start, stop)$$

which returns a match error score after time alignment is used in the minimisation of

$$DP(t_m, a, i)$$

where i ranges over the path restriction neighbourhood from $t_m(l)-k$ to $t_m(l)+k$, for each successive matching of t_m where t_m is a member of the class of permissible templates. It can be seen that the path restriction method is unsuitable due to to the amount of computation for large neighbourhood sizes and for long sequences of templates. However, this method is adequate for a very limited grammar of the type proposed here , and for a restricted neighbourhood size of the order of the DP adjustment window size.

6.2 Nuclear tone analysis and classification

Single template analysis

The SDA system was used in a single template analysis and classification experiment. This was equivalent to using a restricted form of syntax in SDA, where the syntax involved is in the choice of primary tone category most appropriate as a description of the test utterance, rather than tone collocation. The addition of a training process to determine the most suitable reference templates transforms the system operation here into a simple recognition experiment. A database of 400 single syllable utterances was used. The system was trained on a selection of samples and then set to classify the remaining samples within the database as belonging to one of the five tone classes. This was repeated for a number of dynamic programming window lengths, for both speaker dependent and independent recognition modes, and for weighted and unweighted dynamic programming distance measures.

Sample population and PFVS selection

The two databases used for training and classification partitioned a set of 400 speech utterances from four male British English speakers. The samples consisted of five repetitions of each of the five Hallidayan tones spoken on each of four syllables. The samples are denoted in the following way in this thesis:

<tone><syllable><repetition>.<subject id.>

where	tone	ranges from 1 to 5
	syllable	ranges from A to D
	repetition	ranges from 1 to 5
	subject	ranges through DL, SD, PT, DM

The samples were taken using the Laryngograph mode of the preprocessor. The subjects were prompted to speak with the specified intonation by asking for the samples to be spoken 'as a statement', 'as a question', 'weakly', 'with reservation', 'with emphasis'. The prompts were based on the semantic function labels used in the intonation perception experiments of section 3.4 as these clearly have relevance to the perception of British English intonation and also allows a direct comparison to be made between perception and production under similar task instructions. The labels were

1. statement
2. question
3. weak statement
4. reservation
5. emphatic statement

The subjects required little prompting to produce natural sounding stretches of utterance. The carrier syllables had to have continuous voicing for most of the extent of the utterance and they were chosen to be neutral and common enough to occur in informal conversation; it was decided that this would give the the best chance of natural intonation being used. The syllables were

1. yes
2. no
3. mmm . .
4. well

The third syllable is interesting in that the segmental level information is non-existent - all the linguistic information is conveyed by the intonation. The fourth syllable also conveys little at the segmental level, its use in informal conversation being almost exclusively restricted to that of intonation carrier. (This is described as a minor sense of *well* by the Shorter Oxford English Dictionary: 'Employed without construction to introduce a remark . . . frequently used merely as a preliminary or resumptive word'.) The first two syllables are representative of utterances that convey meaning at the segmental level; direct propositional opposites were chosen in order to cancel out any semantic gesture implicit in the carrier syllables. The speakers reported no great difficulty in producing acceptable utterances for any of the carriers despite the range and disparity of the types and functions of the syllables.

The relevant sections of the preprocessed data were selected by an automatic routine. Any automatic selection is a compromise, and as such it is inevitable that some of the PFVS were not as good as if they been selected individually. Against that criticism, it can be held that manual selection involves the constant resetting of the selection criteria; this is inappropriate for an automatic recogniser. The criteria decided on used a smoothed copy of the CF0 measure of the PFVS to decide on the region of high confidence corresponding to good estimates of F0. The smoothing was average-of-four and the threshold was 30 on the self-normalised CF0 scale ranging from 0 to 100. All estimates within the first points of exceeding and falling below the threshold were assumed to be the desired region. The unsmoothed CF0 values were used in the subsequent analysis and classification. Preliminary tests showed that these parameters yielded acceptable results. Figures 5.2 to 5.6 show the bounds selected in this way; inspection of the plots suggests that this method would be hard to better, even by manual selection.

Classifier training and performance

The classifier was trained by selecting a set of five reference templates, each of which minimised the sum of the intra-class distance for the respective classes using the dynamic time warping error as a measure of sample-to-sample distance. In practice, for any one sample this involved the summing of all the scores for matching with all other training samples in its class. Five reference templates were chosen in this way for the classifier every time a change was made in one the SDA system parameters. The selection for each class was on eight samples for the speaker dependent mode; tables 6.1 to 6.4 give the reference templates selected for a range of adjustment window lengths for the four subjects. It can be seen that there is little variation over the adjustment window parameter for all the subjects, and what differences in selection exist are found on the extrema of the window range (indicated by italics in the tables). This suggests there is a steady-state region for r=3 to r=4 in respect to speaker dependent training; it can be seen from the classification experiments below that there is no measurable improvement in the recognition scores for r>3.

The selection for each class for the independent mode was on 33 samples (including the one synthetic PFVS described in section 3.4).

r	*1*	*2*	*3*	*4*	*5*
1	*1B2*	*2D2*	3D2	4B1	5B1
2	*1B2*	*2D2*	3B2	4B1	5B1
3	1D1	2C1	3B2	4B1	5B1
4	1D1	2C1	3B2	4B1	5B1
5	*1C1*	2C1	3B2	4B1	5B1
6	*1C1*	2C1	3B2	4B1	5B1

Table 6.1 *Reference templates selected for speaker 1*

r	*1*	*2*	*3*	*4*	*5*
1	1B2	2B2	3A2	4D2	5D2
2	1B2	2B2	3A2	4D2	5D2
3	1B2	2B2	3A2	4D2	5D2
4	1B2	2B2	3A2	4D2	5D2
5	1B2	2B2	3A2	4D2	5D2
6	*1A1*	2B2	3A2	4D2	5D2

Table 6.2 *Reference templates selected for speaker 2*

r	*1*	*2*	*3*	*4*	*5*
1	1B2	2B1	3B2	4A2	5B1
2	1B2	2B1	3B2	4A2	5B1
3	1B2	2B1	3B2	4A2	5B1
4	1B2	2B1	3B2	4A2	5B1
5	1B2	2B1	3B2	4A2	5B1
6	1B2	*2D1*	3B2	4A2	5B1

Table 6.3 *Reference templates selected for speaker 3*

r	*1*	*2*	*3*	*4*	*5*
1	*1D1*	*2D2*	3D1	4D2	*5B1*
2	1A2	*2D1*	3D1	4D2	5D2
3	1A2	2B2	3D1	4D2	5D2
4	1A2	2B2	3D1	4D2	5D2
5	1A2	2B2	3D1	4D2	5D2
6	1A2	2B2	3D1	4D2	*5B1*

Table 6.4 *Reference templates selected for speaker 4*

r	*1*	*2*	*3*	*4*	*5*
1	*1D1*	*2D2*	3D1	4D2	*5B1*
2	1A2	*2D1*	3D1	4D2	5D2
3	1A2	2B2	3D1	4D2	5D2
4	1A2	2B2	3D1	4D2	5D2
5	1A2	2B2	3D1	4D2	5D2
6	1A2	2B2	3D1	4D2	*5B1*

Table 6.4 *Reference templates selected for speaker 4*

tone	*weighted*	*unweighted*
1	1B2.DL	1B2.DL
2	2C2.PT	2C2.PT
3	3C1.DM	3C1.DM
4	4D2.DM	4D2.DM
5	5B1.DL	5B1.DL

Table 6.5 *Reference templates selected speaker independent mode*

The reference templates for r=3 are given in table 6.5. The speaker dependent training (and the classification, below) appears to stabilise at around the window length of 3 so it was decided to train the independent mode for that value. (The corresponding exhaustive trial over r would have required several hundred hours of computer time.)

The classification of the test samples was achieved by calculating the DTW distance from each of the five reference samples and choosing the minimum as indicating class membership.

Speaker dependent recognition. The effect of varying the adjustment window length on classification performance is given in table 6.6; this is the overall performance averaged over the five tones, each score represents the classifier performance for 60 test samples. It can be seen that there is a slight, but consistent, improvement from r=1 to r=3, thereafter levelling out. The performance for both subjects can be described as good in comparison with other speech recognisers, considering that this system operates on basic units of speech with no adaption or access to other knowledge sources. The breakdown for performance within tone categories is given in table 6.7; similar trends can be observed as with the averaged performance, no class excepted, but with three classes, one, two and four achieving a consistently higher score.

Speaker independent recognition. The classification scores for the five tones using the speaker independent mode of operation is given in table 6.8, each score the result of 240 classifications. The tabled scores are for a window length of 3, for both CF0 weighted and unweighted DTW distance measure. The scores are similar, perhaps showing that the CF0 weight as a confidence measure on the F0 estimate is not a significant factor when the F0 is continuous. For all the selected PFVS used as samples in these experiments, the F0 had a high confidence

r	*subject*	
	1	*2*
1	79.2	76.2
2	88.1	75.0
3	89.9	76.6
4	89.9	76.6

Table 6.6 *Effect of window length on speaker dependent recognition scores*

subject	*r*	*tone* 1	2	3	4	5
1	1	100.0	91.7	83.3	25.0	91.7
	2	100.0	91.7	72.7	83.3	91.7
	3	100.0	91.7	72.7	83.3	91.7
	4	100.0	91.7	72.7	83.3	91.7
2	1	100.0	91.7	75.0	58.3	50.0
	2	100.0	91.7	75.0	58.3	50.0
	3	100.0	100.0	75.0	58.3	50.0
	4	100.0	100.0	75.0	58.3	50.0

Table 6.7 *Speaker dependent recognition scores*

	weighting	
tone	*yes*	*no*
1	56.2	56.2
2	70.8	75.0
3	63.8	63.8
4	64.6	64.6
5	60.4	66.7

Table 6.8 *Speaker independent recognition scores (overall)*

subject	tone 1	2	3	4	5
1	100.0	41.7	63.6	91.7	75.0
2	83.3	100.0	100.0	75.0	41.6
3	41.7	66.7	25.0	41.7	83.3
4	0.0	75.0	66.7	50.0	41.6
reference	1B2.DL	2C2.PT	3C1.DM	4D2.DM	5B1.DL

Table 6.9 *Speaker independent recognition scores (weighted)*

subject	*tone* 1	2	3	4	5
1	100.0	41.7	63.6	91.7	75.0
2	83.3	100.0	100.0	75.0	50.0
3	41.7	66.7	25.0	41.7	83.3
4	0.0	75.0	66.7	50.0	58.3
reference	1B2.DL	2C2.PT	3C1.DM	4D2.DM	5B1.DL

Table 6.10 *Speaker independent recognition scores (unweighted)*

measure. The scores are generally a good deal lower that those for the dependent mode; this was to be expected. A breakdown of these scores across the four speakers is given in tables 6.9 (weighted) and 6.10 (unweighted) together with the identification of the selected reference templates. The scores here show less consistency, the tones from some speakers completely misclassified throughout, and the tones from the subject providing the reference template achieving higher than average classification. The overall scores for all speakers and all tones were 63.2% (weighted) and 64.4% (unweighted); that compares with the overall score for speaker dependent classification of 83.2%

6.3 Connected utterance performance

The SDA analyser was run in the connected analysis mode described in section 6.1 as an indication of how template collocation could be handled in automatic recognition. Samples of the two parts of the minimal pair analogy of table 3.1 were digitised and preprocessed as described above. The sample duration was two seconds and the initial lead-in of silence was excluded to within 50 ms of the first F0 trace as determined by the automatic selection procedure of 6.2. The utterances were

Left for Italy.

Left for *It*aly?

with the nuclear F0 movement expected on the first syllable of *Italy*. Figures 6.1 and 6.2 show the two PFVS samples in the same format as the single template graphs of figures 5.2 to 5.6, the time axis is annotated to show the occurrence of the segmental phonemes. The graphs are more noisy than the single template examples, as expected, as there are periods of unvoiced speech present; however, the high CF0 regions which correspond to the syllabic voicing peaks show F0 traces that are well-formed and consistent. The traces for the last syllable are interesting as these display the characteristic fall/rise of the tone group nucleus observed in the single template recognition experiments.

The constraining grammar C.SYN used here is given in full in appendix A3. It partitions the tone group into two obligatory elements:

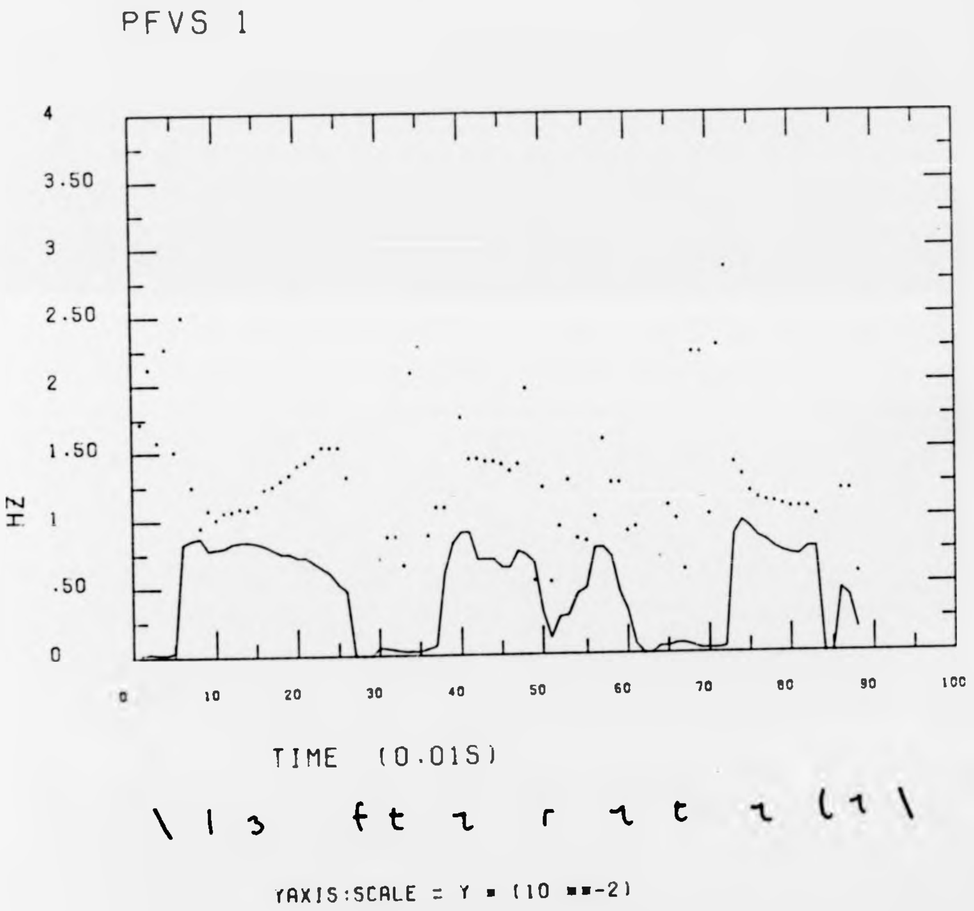


Figure 6.1 *First connected example PFVS*

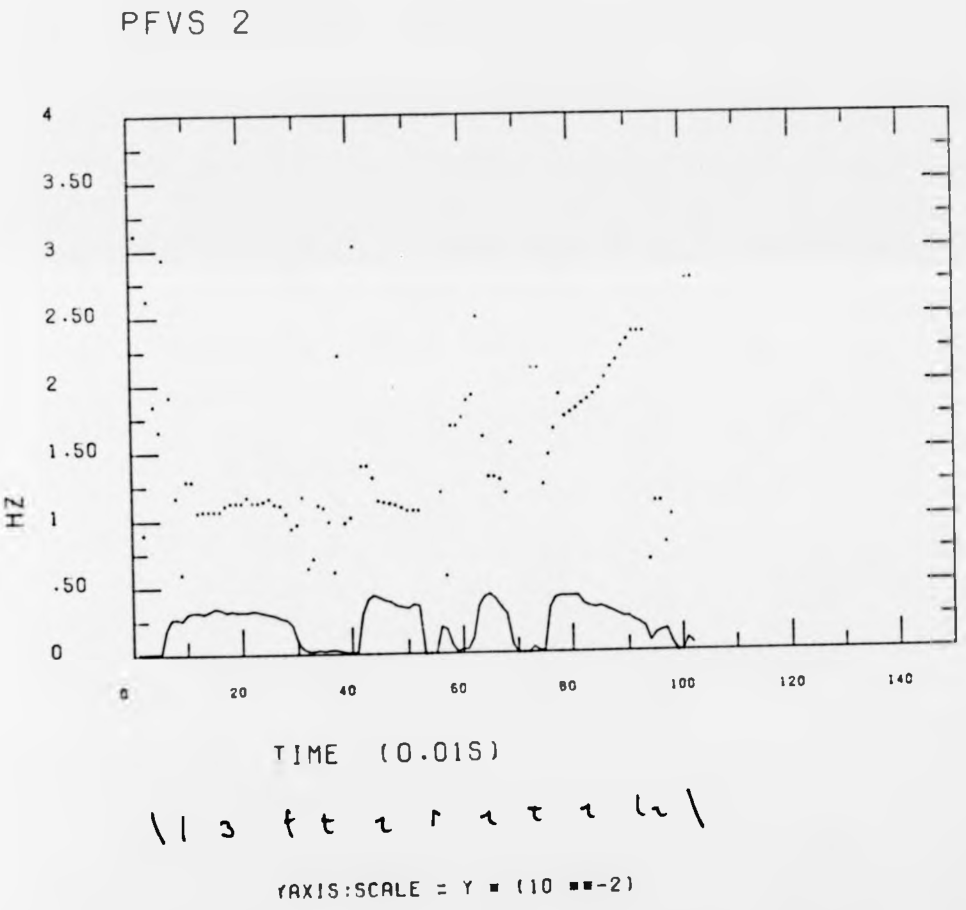


Figure 6.2 *Second connected example PFVS*

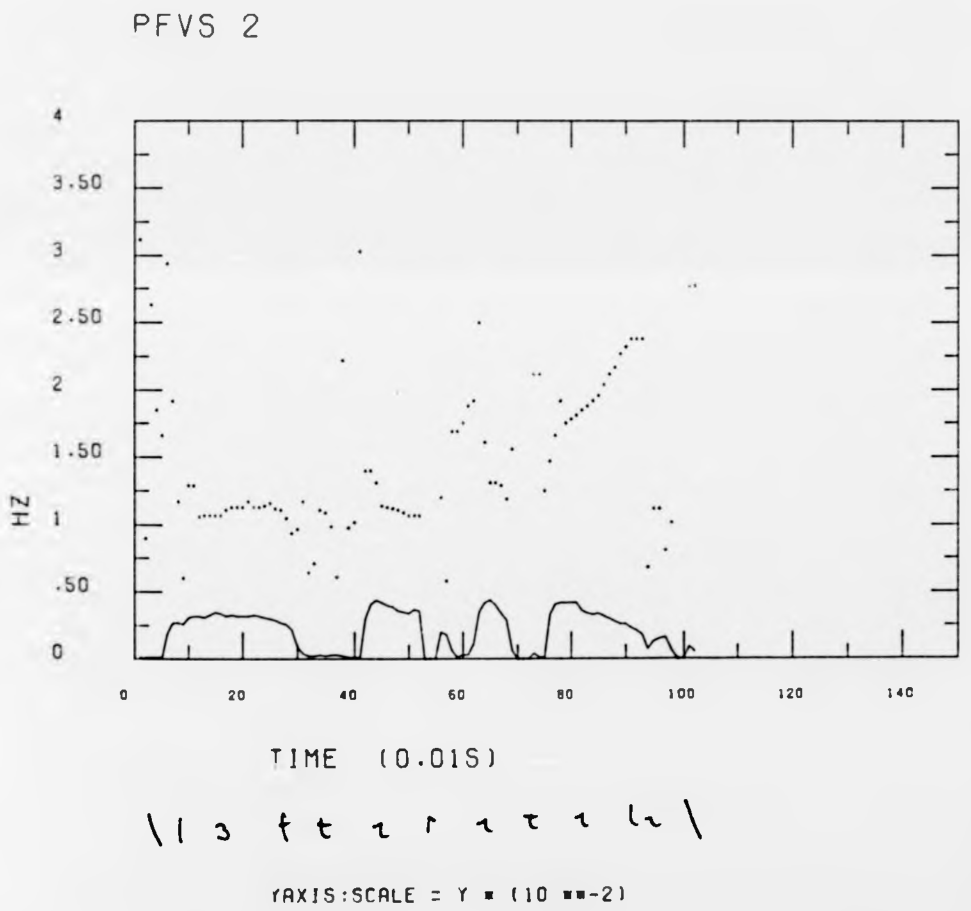


Figure 6.2 *Second connected example PFVS*

1. *Head*. The phonetic descriptions of tone group elements apart from the nucleus are vague or merely unavailable. The *head* element expected here is no exception: the literature suggests that the head can be analysed as spanning the discontinuous F0 traces of the prenuclear syllables (from the first stressed syllable) to give a flat, extensible F0 contour. The grammar uses a template, T HEAD, taken from the PFVS of the first utterance (statement) using the selectional criteria described. This template was used for both examples as the first (obligatory) template; the *left for* PFVS of the statement was matched against the first section of the full statement and the full question.
2. *Nucleus*. The grammar describes the nucleus as being a choice from the five Hallidayan simple tones. The templates were those used in the speaker independent recognition experiments for adjustment window size r=2. Essentially, the selection of the nucleus template within the full analysis run can be regarded as analogous to the single template classification experiments of the preceding section.

The SDA system generates results files for each analysis run; the files for the two PFVS samples are given in tables 6.11 and 6.12. Several points can be made:

```
Syntax Directed Prosodic Analysis

intermediate code from c1               .A and .C
speech data from       c1.dl
results to             c1               .RES

Analysis returns match error 4.803128E+00 for     0.93 seconds of
input.

The sequence of terminal symbols achieving that score is

t head
t tone 1


The phrase structure as analysed by the grammar is
```

$[_{\text{sigma}}[_{\text{pre head}}][_{\text{head}}\ \text{t head}\ ][_{\text{nucleus}}\ \text{t tone 1}\ ]]$

Table 6.11 *Computer generated results for first connected run*

```
Syntax Directed Prosodic Analysis

intermediate code from c2              .A  and .C
speech data from       c2.dl
results to             c2              .RES

Analysis returns match error 7.473013E+00 for     0.88 seconds of
input.

The sequence of terminal symbols achieving that score is

t head
t tone 2


The phrase structure as analysed by the grammar is
```

$[_{\text{sigma}}[_{\text{pre head}}][_{\text{head}}\ \text{t head}\ ][_{\text{nucleus}}\ \text{t tone 2}\ ]]$

Table 6.12 *Computer generated results for second connected run*

	PFVS	
tone	*1*	*2*
1	4.803	10.03
2	6.552	7.473
3	5.190	9.648
4	5.741	9.291
5	7.482	9.145

Table 6.13 *Internal decision values of connected runs*

1. The appropriate templates were chosen correctly for the two samples. The best candidate scores for each of the five nuclear tones during the matching for both PFVS samples are given in table 6.13. (These values were recovered from debugging information from the analysis program after setting an internal program flag.)

2. Apart from displaying the correct choice, they show that, for templates, tone 3 was considered to be the nearest to tone 1 and that tone 4 was considered the nearest to tone 2. Tones 1, 2 and 5 fall into one natural semantic class: statement (ordinary, weak and emphatic); tones 2 and 4 fall into another: question (ordinary and reserved). The selection of the second best here would not result in a serious misinterpretation of semantic gesture. It appears that the system behaves as we would expect a listener to behave in this respect.

3. The use of a section of the first sample as a template for assisting matching the second sample appears not to have degraded the subsequent classification of the nuclear tone. This suggests that using a stylised description of the head may be worthwhile.

4. It should also be noticed that the nuclear templates are those used for the speaker *in*dependent classification, where the overall performance was a fair, but not outstanding, 63.2%. We may assume that the connected recognition task will be more difficult.

The conclusion can be drawn from the performance of the SDA system in these two runs that connected intonation analysis may be practicable.

CHAPTER 7

DISCUSSION

Intonation description as ASR constraint

Lea (1980) presented an review of prosody in automatic speech recognition and gives a substantial catalogue of prosodic features amenable to analysis and an account of several such analysis algorithms, but there is no speech recognition system presented, or even a proposal for one. Prosodic cues are discussed at several levels of language representation and of the recognition process:

1. Independent access to acoustic events, by-passing the classical analysis into words
2. Resolution of ambiguities in text and purely word-sequence analyses, providing structural bracketing for an utterance
3. Sentence type identification, and sentence inclusion (subordination and coordination)

Lea also discusses two approaches for use in automatic speech recognition. The first is to use this prosodic information as a check on an existing analysis, as an independent though secondary system. An example given is the reassignment of likelihood scores to competing hypotheses; candidate words are given a stronger weight if an expected F0 fall-rise does in fact occur. The second method is to take the prosodic analysis as being of equal importance to the segmental and lexical analysis, in effect creating a parallel recognition process with its own access to acoustic events and its own syntax. Lea summarises the current position on prosodic analysis in speech recognition by suggesting that testing the use of prosodics by means of a prosodics-only recogniser (Lea 1980, p201).

The SDA system described in this thesis is just such a recogniser. In the course of bringing together and operating the complete system, several conclusions have been reached:

1. Intonation phenomena is amenable to linguistic description
2. Such a description can be used in constraining speech recognition
3. The basic system of nuclear syllable intonation is more complex than a binary fall/rise opposition; the nuclear intonation typology should deal with about five tones, closely modelled on Halliday's scheme
4. Classification performance is generally insensitive to DTW adjustment window variations after some DTW.
5. The pitch movement on tonetic syllables is perceived categorically
6. Continuous utterance prosodic analysis is possible to a limited extent

The SDA system may be considered an extension of current speech recognition systems in two major respects:

1. As it stands, it is a prosodics-only recogniser in the sense of Lea's comment above. In that respect it has been used to recognise the presence of formal patterns in the prosodic features of speech.
2. In its heuristic two-level dynamic programming template matching it approaches the operation of current connected word recognition strategies described in the literature.

With regard to the second point, this system is closely equivalent to these other systems in the matching algorithms and in the concept of a two-level matching process. However, those systems have made no explicit reference to the class of syntactic constraints imposed upon the selection of templates in recognition (Sakoe 1979), or have chosen to use a weak class of grammars such as finite-state grammars to handle language phenomena (Bridle 1983). The SDA system currently employs a class of grammars which are of the context-free, phrase structure type of which finite-state grammars are a special class.

Future directions

Proposals for further work include:

Continuation of the prosodics-only analysis described here, but with the introduction of speaker normalisation in the frequency dimension. This would involve a two-dimensional dynamic programming algorithm to non-linearly warp both the time scale and the frequency scale of the matched templates. Such as scheme as been proposed for spectral templates (Moore 1979) and implemented in isolated word recognition (Paliwal and Ainsworth 1984). In the case of frequency spectrum representations of speech the information contained in the frequency dimension is linguistically significant; with intonation there may be less linguistic information contained in the absolute F0 values of a section of contour. A possible hypothesis is that a matching algorithm would therefore align frequency values in template and data, effectively normalising the data, without distorting the phonological content as may happen with frequency spectra. The implementation cost of such a scheme would be minimal as the representation frames in question are single values.

Adding the F0 and CF0 information to an existing recognition algorithm which uses only frequency spectrum information. A simple scheme could be used where the the F0 value (if present) is used as a weight on the spectral frame distance in direct proportion to its value. The notion behind this is that regions of F0 peak will correspond to regions of unreduced vowels in connected utterance which will match single word templates more directly.

Using the SDA system to handle spectral templates as well the F0 templates used now. The SDA analyser could be modified to include concurrent specification - this would be a rule type specifying that another item of the grammar is to be recognised starting at the same portion of the speech data. The present system can only specify that an item can occur in the speech data after, or instead of, another item. A simple modification to the SDA system would allow multiple analyses of the same data to occur, each directed by its own grammar. In effect this would be pseudo-parallel recognition controlled by several knowledge sources. (All that would have to be changed in the analyser is that the current position along the preprocessed speech would not be advanced to the 'next' item - otherwise the analysis would proceed as for a concatenation of items. A concurrent rule example could be:

integrated grammar = (word level grammar,tone grammar);

where the recognition score for the total rule would simply be the sum of the two scores, each recognition taking place sequentially, but from the same position.

The aim of this thesis was to investigate the use of knowledge of the British English intonation system to constrain speech recognition. A prosodics-only recogniser was constructed to analyse intonational phenomena. The recogniser was constrained by a linguistic description of the British English nuclear tones. The description was obtained from experiments on the categorical nature of the perception of the pitch movement on tonic syllables. The recogniser was successful in both speaker dependent and independent classification tasks and shows promise for connected utterance analysis.

REFERENCES

Abercrombie, D (1967). *Elements of General Phonetics*, Edinburgh: Edinburgh University Press.

Ainsworth, W A (1974). 'Performance of a speech synthesis system', *International Journal Man-machine Studies* 6, 493-511.

Ainsworth W A, Lindsay D (1984). 'Perception of pitch movement on tonic syllables in British English' (submitted to *Journal of the Acoustical Society of America*).

Armstrong, L E, Ward, I C (1926). *Handbook of English Intonation*, Cambridge: Heffer.

Ashby, M G (1978). 'A study of two English nuclear tones', *Language and Speech* 21 (4), 326-336.

Austin, J L (1962). *How To Do Things With Words*, The William James Lectures, Harvard University, 1955, (ed. J O Urmson), London: Oxford University Press.

Bauer, F L, Eickel, J (eds.) (1974). *Compiler Construction : An Advanced Course*, New York: Springer-Verlag.

Beaugrand, R de, Dressler, W (1982). *Introduction To Text Linguistics*, London: Longman.

Bellman R E (1957). *Dynamic Programming*, Princetown University Press.

Bloomfield, (1933). *Language*, New York: Holt.

Bolinger, D L (1951). 'Intonation - levels *vs*. configuration', *Word* 7, 199-210.

Bolinger, D L (1961). *Generality, Gradience and the All-or-None*, The Hague: Mouton.

Bornat, R (1979). *Understanding and Writing Compilers*, London: Macmillan.

Bridle, J S (1973). 'An efficient elastic-template method for detecting given words in running speech', *British Acoustical Society Meeting*, April 1973, 1-4.

Bridle, J S, Brown, M D, Chamberlin, R M (1983). 'Continuous connected word recognition using whole word templates', *The Radio and Electronic Engineer* 53 (4), 167-175.

Charniak, E, and Wilks, Y (1976). *Computational Semantics*, Amsterdam: North-Holland.

Chomsky, N (1957). *Syntactic Structures*, The Hague: Mouton.

Chomsky, N (1965). *Aspects of the Theory of Syntax*, Cambridge, Mass.: MIT Press.

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

Cohen, A, and 't Hart, J (1967). 'On the anatomy of intonation', *Lingua* 19, 177-192.

Cooper, W E, and Sorenson, J M (1981). *Fundamental Frequency in Sentence Production*, Berlin: Springer-Verlag.

Crompton, A S (1980). 'Intonation: Tonetic stress marks *versus* levels *versus* configurations', *Word* 31, 151-198.

Crystal, D (1969). *Intonation and Prosodic Systems in English*, Cambridge: Cambridge University Press.

Crystal, D (1969b). Review of Halliday 1967, in *Language* 45 (2), 378-394.

Crystal, D, Quirk, R (1964). *Systems of Prosodic and Paralinguistic Features in English*, The Hague: Mouton.

Currie, K L (1980). 'An initial search for "tonics"', *Language and Speech* 23 (4), 329-350.

Cutting J E, Rosner B S (1974). 'Categories and boundaries in music and speech', *Perception and Psychophysics* 16 (3), 564-570.

Davis, K H, Biddulph, R, Balashek, S (1952). 'Automatic recognition of spoken digits', *Journal of the Acoustical Society of America* 24, 637-642.

DeMori, R (1977). 'Syntactic recognition of speech patterns', in K S Fu 1977.

Denes, P (1959). 'A preliminary investigation of certain aspects of intonation', *Language and Speech* 2, 106-122.

Durling, A E (1964). *Computational Aspects of Dynamic Programming in Higher Dimensions*, PhD Thesis, Syrcacuse University.

Edward, J A (1982). 'Rules for synthesising the prosodic features of speech', *JSRU Research Report No. 1015*, Cheltenham: Joint Speech Research Unit.

Fourcin, A J, Abberton, E (1971). 'First applications of a new laryngograph', *Medical and Biological Illustration* 21, 172-182.

Fonagy, I (1978). 'A new method of investigating the perception of prosodic features', *Language and Speech* 21 (1), 34-49.

Friedman J (1976). *A Computer Model of Transformational Grammar*, New York: Elsevier.

Fry D B (1958). 'Experiments in the perception of stress', *Language and Speech* 1, 126-152.

Fu, K S (1974). *Syntactic Methods in Pattern Recognition*, New York: Academic Press.

Fu, K S (1977). *Syntactic Pattern Recognition*, Berlin : Springer-Verlag.

Green, N (1972). 'Automatic speaker recognition using pitch measurements in conversational speech', *JSRU Research Report No. 1000*, Cheltenham: Joint Speech Research Unit.

Greene, J (1971). *Psycholinquistics*, Harmondsworth England: Penguin.

Griffiths, M (1974). 'LL(1) grammars and analysers' in Bauer and Eickel 1974.

Halliday, M A K (1961). 'Categories of the theory of grammar', *Word*

17, 241-292.

Halliday, M A K (1963). 'The tones of English', *Archivum Linguisticum* 15, 1-28.

Halliday, M A K (1967). *Intonation and Grammar in British English*, The Hague: Mouton.

Hart, J (1569). *The opening of the unreasonable writing of our inglish toung*. (in B Danielsson 1955, ed. *Works on English Orthography and Pronunciation; 1551, 1569, 1570*, Stockholm: Almqvist and Wiksell.)

t'Hart, J (1979). 'Explorations in automatic stylization of F0 curves', *IPO Annual Progress Report* 14, 61-66.

t' Hart, J, Cohen, A (1973). 'Intonation by rule: A perceptual quest', *Journal of Phonetics* 1, 309-328.

Hess, W J (1983). *Pitch Determination of Speech Signals*, Berlin: Springer-Verlag.

Hjelmslev, L (1961). *Prolegomena to a Theory of Language*, translated by F J Whitfield, Madison: University of Wisconsin Press. (Original title *Omkring Sprogteoriens Grundloeggelse*, Copenhagen: Munksgaard, 1943.)

Hocket, C F (1958). *A Course in Modern Linguistics*, New York: Macmillan.

Hocket, C F (1960). 'The origin of speech', *Scientific American* 203, 89-96.

Hocket, C F, Altmann, S (1968). 'A note on design features', in *Animal Communication*, T A Sebeok, ed., 61-72, Bloomington, Ind. : Indiana University Press.

Hunt, M J (1978). 'Automatic correction of low-frequency phase distortion in analogue magnetic recordings', *Acoustic Letters* 2, 6-10.

Iles, L A (1967). 'M A K Halliday's "Tones of English" in synthetic speech', *Work in Progress* 1, Phonetics Department, Edinburgh University.

Jassem, W (1978). 'On the distributional analysis of pitch phenomena', *Language and Speech* 21 (4), 362-372.

Jensen, K, Wirth N (1974). *User Manual and Report*, New York: Springer-Verlag.

Kingdon, R (1958). *The Groundwork of English Intonation*, London: Longman.

Klatt, D H (1977). 'Review of the ARPA speech understanding project', *Journal of the Acoustical Society of America* 62, 1345-1366.

Knuth, D E (1971). 'Top-down syntax analysis', *Acta Informatica* 1, 79-110.

Ladd, D R (1978). 'Stylized intonation', *Language* 54 (3), 517-540.

Lamb, S M (1966). *Outline of Stratificational Grammar*, Washington D.C.: Georgetown University Press.

Lea, W A (1980a). *Trends in Speech Recognition*, New Jersey: Prentice-Hall.

Lea, W A (1980b). 'Prosodic aids to speech recognition', in Lea 1980a, 166-205.

Lea, W A, Medress, M F, and Skinner, T E (1975). 'A prosodically guided speech understanding system', *IEEE Transactions* ASSP-23, 30-38.

Lea, W A, Shoup, J E (1980). 'Specific contributions of the ARPA SUR project', in Lea 1980a, 382-421.

Lehiste, I (1970). *Suprasegmentals*, Cambridge, Mass. : MIT Press.

Leith, P (1983). 'Hierarchically structured production rules', *The Computer Journal* **26** (1), 1-5.

Lewis, P M, Stearns, R E (1968). 'Syntax-directed transduction', *Journal of the Association for Computing Machinery* 15 (3), 465-488.

Liberman, A M, Harris, K S, Hoffman, H S, Griffith, B C (1957). 'The discrimination of speech sounds within and across phoneme boundaries', *Journal of Experimental Psychology* 54, 358-368.

Liberman, A M, Harris, K S, Kinney, J A, Lane, H (1961). 'The discrimination of relative onset time of the components of certain speech and nonspeech patterns', *Journal of Experimental Psychology* 61, 379-388.

Lieberman, P (1967). *Intonation, Perception and Language*, Cambridge, Mass.: MIT Press.

Lieberman, P (1968). 'Primate vocalisations and human linguistic ability', *Journal of the Acoustical Society of America* 44, 1574-1684.

Lieberman, P (1981). 'On the evolution of human speech', in Myers *et al*. eds. 1981, 271-280.

Lindsay, D (1983). 'A method of describing pitch phenomena', in *Investigations of the Speech Process* (ed. P Winkler), 189-210, Bochum: Studienverlag Brockmayer.

Lindsay, D, Ainsworth, W A (1982). 'The computer analysis of prosody', *Proceedings of the 1982 Autumn Conference of the Institute of Acoustics*, E2.

Lockwood, D G (1972). *Introduction to Stratificational Linquistics*, New York: Hardcourt Brace Jovanovitch.

Lyons, J (1977) *Semantics* 1, Cambridge: Cambridge University Press.

Macmillan, N A, Kaplan, H L, Creelman, C D (1977). 'The psychophysics of categorical perception', *Psychological Review* 34, 452-471.

Marcus, S M (1981). 'ERIS - context sensitive coding in speech perception', *Journal of Phonetics* 9, 197-220.

Markel, J D, Gray A H (1976). *Linear Prediction of Speech*, Berlin: Springer-Verlag.

Martin, T B (1980). 'Practical speech recognisers and some performance effectiveness parameters', in Lea 1980a, 24-38.

Martinet, A (1949). 'La double articulation linguistique', *Travaux du Cercle Linguistique de Copenhague* 5, 30-37.

Mattingly, I G (1966). 'Synthesis by rule of prosodic features', *Language and Speech* 9, 1-13.

Miller G A (1956). 'The magical number seven plus or minus two, or, some limits on our capacity for processing information', *Psychological Review* 63, 81-96.

Miller, J D, Wier, C C, Pastore, R E, Kelly, W J, Dooling, RJ (1976). 'Discrimination and labeling or noisebuzz sequences with varying noise-lead times: An example of categorical perception', *Journal of the Acoustical Society of America* 60, 410-417.

Moore, R K (1979). 'A dynamic programming algorithm for the distance between two finite areas', *IEEE Transactions* **PAMI-1** (1), 86-88.

Myers, C S, Rabiner, L R (1981). 'A level-building dynamic time warping algorithm for connected word recognition', *IEEE Transactions* **ASSP-29** (2), 284-297.

Myers, T, Laver, J, Anderson, J (eds.) (1981). *The Cognitive Representation of Speech*, Amsterdam: North-Holland.

Newell, A, Barnett, J, Forgie, J, Green, C, Klatt, D, Licklinder, J C R, Munson, J, Reddy, D R, Woods, W (1971 and 1973). *Speech Understanding Systems*, Final report of study group, Amsterdam: North-Holland.

Noll, A M (1964). 'Short-time spectrum and "cepstrum " techniques for vocal pitch detection', *Journal of the Acoustical Society of America* **36** (2), 296-302.

Norman, J M (1972). *Heuristic Procedures in Dynamic Programming*, Manchester: Manchester University Press.

O'Connor, J D, Arnold, G F (1961). *Intonation of Colloquial English*, London: Longman.

Ohman, S E G (1967). 'Word and sentence intonation: A quantitative model', *Speech Transmission Laboratory, Quarterly Progress Report* 2-3 (Stockholm: Royal Institute of Technology) 20-54.

O'Shaughnessy, D (1979). 'Linguistic patterns in fundamental frequency', *Journal of Phonetics* 7, 119-145.

Paliwal, K K, Ainsworth, W A (1984). 'Dynamic frequency warping for speaker adaptation in automatic speech recognition', submitted to *Speech Communication*.

Paliwal, K K, Agarwal, A, Sinha, S S (1982). 'A modification over Sakoe and Chiba's dynamic time warping algorithm for isolated word recognition', *Signal Processing* 4, 329-333.

Paliwal, K K, Lindsay, D, Ainsworth, W A (1982). 'Correlation between production and perception of English vowels', *Journal of Phonetics* 11, 77-83.

Palmer, H E (1922). *English Intonation*, Cambridge: Heffer.

Peckham, J B, Green, J R D, Canning, J V, and Stephens, P (1982). 'A real time hardware continuous speech recognition system', *IEEE Transactions* ASSP, 863-866.

Pierrehumbert, J (1979). 'The perception of fundamental frequency declination', *Journal of the Acoustical Society of America* 66 (2), 363-369.

Pierrehumbert, J (1980). *The Phonetics and Phonology of Intonation*, PhD Thesis, MIT.

Pierrehumbert, J (1981). 'Synthesizing intonation', *Journal of the Acoustical Society of America* 70 (4), 985-995.

Pijper, J R de (1979). 'Close-copy stylisations of British English intonation contours', *IPO Annual Progress Report* 14, 66-71.

Pilch, H (1980). 'English intonation as phonological structure', *Word* 31, 55-67.

Ritchie, G D (1977). *Computer Modelling of English Grammar*, PhD thesis, Department of Computer Science, University of Edinburgh.

Rosen, S M, Howell P (1981). 'Plucks and bows are not categorically perceived', *Perception and Psychophysics* 30 (20), 156-168.

Sakoe, H (1979). 'Two-level DP algorithm - A dynamic programming based pattern matching algorithm for connected word recognition', *IEEE Transactions* **ASSP-27** (6), 588-595.

Sakoe, H, Chiba, S (1978). 'Dynamic programming algorithm optimisation for spoken word recognition', *IEEE Transactions* ASSP-**26** (1), 43-49.

Sapir, E (1925). 'Sound patterns in language', *Language* 1, 37-51.

Saussure, F de (1959). *Course in General Linguistics*, Glasgow: Fontana/Collins. (Translated from the French edition of Bally and Sechehaye of lecture courses 1906-1911.)

Schane, S A (1974). *Generative Phonology*, New Jersey: Prentice-Hall.

Shannon, C E, Weaver, W (1949). *The Mathematical Theory of Communication*, Urbana, Illinois: University of Illinois Press.

Sommerstein, A H (1977). *Modern Phonology*, London: Edward Arnold.

Stevens, K N (1975). 'Quantal configurations of vowels', *Journal of the Acoustical Society of America* 57, (S1) 570(A).

Stewart, A H (1976). *Graphical Representation of Models in Linguistic Theory*, Bloomington, Ind.: Indiana University Press.

Studdert-Kennedy, M, Hadding, K (1973). 'Auditory and linguistic processes in the perception of intonation contours', *Language and Speech* **16**, 293-313.

Trager, G L, Smith, H L (1957). *An Outline of English Structure*, from *Studies in Linguistics: Occasional Papers* 3, Washington:

American Council of Learned Societies.

Trubetzkoy, N S (1933). *Grundzuge der Phonologie*, Prague: Cercle Linguistique de Prague.

Vaissiere, J (1974). 'On French prosody', *MIT RLE Quarterly Progress Report* 114, 212-223.

Vaissiere, J (1975). 'Further note on French prosody', *MIT RLE Quarterly Progress Report* 115, 251-262.

Vaissiere, J (1977). 'La structuration acoustique de la phrase francaise', *Annali della Scuola Normale Superiore di Pisa*, Series III, X (2), 530-560.

Vaissiere, J (1980). 'The search for language-independent prosodic features', First International Conference on the Perception of Speech, Florence, December, 1980.

Vaissiere, J (1981). 'Speech recognition programs as models of speech perception', in *The Cognitive Representation of Speech* (eds. T Myers, J Laver, J Anderson), 443-458, Amsterdam: North-Holland.

van Wijngaarden, A (1974). *Revised Report on the Algorithmic Language Algol 68*, Berlin: Springer-Verlag.

Velichko, Z M, Zagoroyko, N G (1970). 'Automatic recognition of 200 words', *International Journal Man-machine Studies* 2 (3), 222-234.

White, G M, Nealy, R B (1976). 'Speech recognition experiments with linear prediction, bandpass filtering and dynamic programming', *IEEE Transactions* ASSP-24, 183-188.

Wilks, Y. (1976). 'Parsing English II', in Charniak and Wilks (eds.), 1976, 155-184.

Winograd, T (1983). *Language as a Cognitive Process I: Syntax*, Reading, Mass.: Addison-Wesley.

Wood, C C (1976). 'Discriminability, response bias, and phoneme categories in discrimination of voice onset time', *Journal of the*

Acoustical Society of America 60, 1381-1389.

Woods, W A (1970). 'Transition network grammars for natural language analysis', *Communications of the ACM* 13 (10), 591-606.

Young, S J, Fallside, F (1979). 'Speech synthesis from concept: A method for speech output from information systems', *Journal of the Acoustical Society of America*, 66 (3), 685-695.

APPENDICES

Appendix 1 One-Level Wan Wijngaarden Grammar Specification of the SDA Driving Grammar (van Wijngaarden 1974)

```
                grammar : entrance declaration, rule sequence.

          rule sequence : rule sequence, rule;
                          rule.

                   rule : rule name, context,
                          becomes symbol, rh part.

                rh part : tactic symbol;
                          terminal symbol.

          tactic symbol : name, concatenation symbol,
                          tactic symbol;
                          name.

        terminal symbol: left square bracket symbol,
                          transform specification,
                          segment specifier,
                          right square bracket symbol.

    transform specifier : transform specifier;
                          transform, go on symbol.

              transform : ordinal specifier,
                          left bracket symbol,
                          minimum extender,
                          go on symbol,
                          maximum extender,
                          right bracket symbol.

      segment specifier : ordinal specifier,
                          left bracket symbol,
                          prosodic feature vector string,
                          right bracket symbol.

prosodic feature vector string :
                          prosodic feature vector,
                          go on symbol,
                          prosodic feature vector string
                          prosodic feature vector.

prosodic feature vector :
                          left bracket symbol,
                          F0 specifier,
                          go on symbol,
                          EF0 specifier,
                          go on symbol,
                          SE specifier,
                          go on symbol,
                          MSE specifier,
                          right bracket symbol.

                   name : letter symbol, identifier extender;
                          letter symbol.
```

```
  identifier symbol : letter symbol;
                       integer symbol.

       real number : signed integer number, pointer, expo;
                     signed integer number, pointer;
                     signed integer number, expo;
                     signed integer number.

           pointer : dot symbol, integer number;

              expo : exponent symbol, signed integer number;

signed integer number :
                     plus symbol, integer number;
                     minus symbol, integer.

    integer number : number symbol, integer number;
                     number symbol.

    becomes symbol : letter = symbol;

left bracket symbol : letter ( symbol.

right bracket symbol : letter ) symbol.

       plus symbol : letter + symbol.

      minus symbol : letter - symbol.

      go on symbol : letter , symbol.

        dot symbol : letter . symbol.

  separator symbol : letter ; symbol.

     number symbol : letter 0 symbol;
                     letter 1 symbol;
                     letter 2 symbol;
                     letter 3 symbol;
                     letter 4 symbol;
                     letter 5 symbol;
                     letter 6 symbol;
                     letter 7 symbol;
                     letter 8 symbol;
                     letter 9 symbol;

     letter symbol : letter a symbol;
                     letter b symbol;
                     letter c symbol;
                     letter d symbol;
                     letter e symbol;
                     letter f symbol;
                     letter g symbol;
                     letter h symbol;
                     letter i symbol;
                     letter j symbol;
                     letter k symbol;
                     letter l symbol;
                     letter m symbol;
                     letter n symbol;
letter o symbol;
letter p symbol;
letter q symbol;
letter r symbol;
letter s symbol;
letter t symbol;
letter u symbol;
letter v symbol;
letter w symbol;
letter x symbol;
letter y symbol;
letter z symbol;
```

Appendix 2 One-Level Wan Wijngaarden Grammar Specification of the SDA Driving Grammar (van Wijngaarden 1974). T-rule extension specification.

```
                grammar : entrance declaration, rule sequence.

          rule sequence : rule sequence, rule;
                          rule.

                   rule : PS rule; T rule;

                 T rule : left square bracket symbol,
                          rule sequence,
                          T becomes symbol,
                          rule sequence,
                          right square bracket symbol.

                PS rule : rule name, context,
                          becomes symbol, rh part.

                rh part : tactic symbol;
                          terminal symbol.

          tactic symbol : name, concatenation symbol,
                          tactic symbol;
                          name.

        terminal symbol:  left square bracket symbol,
                          transform specification,
                          segment specifier,
                          right square bracket symbol.

    transform specifier : transform specifier;
                          transform, go on symbol.

              transform : ordinal specifier,
                          left bracket symbol,
                          minimum extender,
                          go on symbol,
                          maximum extender,
                          right bracket symbol.

      segment specifier : ordinal specifier,
                          left bracket symbol,
                          prosodic feature vector string,
                          right bracket symbol.

prosodic feature vector string :
                          prosodic feature vector,
                          go on symbol,
                          prosodic feature vector string
                          prosodic feature vector.

prosodic feature vector :
                          left bracket symbol,
                          F0 specifier,
                          go on symbol,
                          EF0 specifier,
                          go on symbol,
                      SE specifier,
                      go on symbol,
                      MSE specifier,
                      right bracket symbol.

               name : letter symbol, identifier extender;
                      letter symbol.

  identifier symbol : letter symbol;
                      integer symbol.

        real number : signed integer number, pointer, expo;
                      signed integer number, pointer;
                      signed integer number, expo;
                      signed integer number.

            pointer : dot symbol, integer number;

               expo : exponent symbol, signed integer number;

signed integer number :
                      plus symbol, integer number;
                      minus symbol, integer.

     integer number : number symbol, integer number;
                      number symbol.

     becomes symbol : letter = symbol;

   T becomes symbol : letter == symbol;

        plus symbol : letter + symbol.

       minus symbol : letter - symbol.

       go on symbol : letter , symbol.

         dot symbol : letter . symbol.

   separator symbol : letter ; symbol.

left bracket symbol : letter ( symbol.

right bracket symbol : letter ) symbol.

left square bracket symbol :
                      letter [ symbol.

right square bracket symbol :
                      letter ] symbol.

      number symbol : letter 0 symbol;
                      letter 1 symbol;
                      letter 2 symbol;
                      letter 3 symbol;
                      letter 4 symbol;
                      letter 5 symbol;
                      letter 6 symbol;
                      letter 7 symbol;
                      letter 8 symbol;
                      letter 9 symbol;

letter symbol :       letter a symbol;
                      letter b symbol;
                      letter c symbol;
                      letter d symbol;
                      letter e symbol;
                      letter f symbol;
                      letter g symbol;
                      letter h symbol;
                      letter i symbol;
                      letter j symbol;
                      letter k symbol;
                      letter l symbol;
                      letter m symbol;
                      letter n symbol;
                      letter o symbol;
                      letter p symbol;
                      letter q symbol;
                      letter r symbol;
                      letter s symbol;
                      letter t symbol;
                      letter u symbol;
                      letter v symbol;
                      letter w symbol;
                      letter x symbol;
                      letter y symbol;
                      letter z symbol;
```

Appendix A3 Intonation grammars used by SDA system

1. Grammar of single template 1A4.SD.
2. Grammar of single template 2A4.SD.
3. Grammar of single synthetic template 1.AV.
4. Grammar for connected utterance example
5. Connected utterance grammar listing from Rule Translator

1. Grammar of single template 1A4.SD.

```
#
    Sigma = Template;
#
    Template = [
#
# dummy match constraints
#
              1(-1,1), 1(-1,1), 1(-1,1), 1(-1,1),
#
# the template proper
#
              (
                161,    44,
                166,    49,
                149,    58,
                149,    65,
                149,    76,
                149,    81,
                144,    84,
                144,    82,
                144,    83,
                140,    88,
                140,    93,
                138,    90,
                135,    92,
                135,    90,
                131,    92,
                125,   100,
                121,    91,
                121,    87,
                119,    83,
                113,    79,
                109,    77,
                105,    84,
                103,    81,
                101,    69,
                 99,    63,
                102,    60,
                100,    56,
                 56,    50,
                            ) ];
```

2. Grammar of single template 2A4.SD.

```
#
     Sigma = Template;
#
     Template = [
#
# dummy match constraints
#
                 1(-1,1), 1(-1,1), 1(-1,1), 1(-1,1),
#
# the template proper
#
                 (
                    232,    45,
                    232,    27,
                      0,     0,
                      0,     0,
                    109,    52,
                    107,    61,
                    126,    83,
                    113,   100,
                    117,    94,
                    119,    91,
                    121,    76,
                    133,    68,
                    135,    56,
                    140,    73,
                    163,    76,
                    169,    73,
                    200,    69,
                    277,    53,
                    303,    42,
                                ) ];
```

3. Grammar of single synthetic template 1.AV.

```
#
    Sigma = Template;
#
    Template = [
#
# dummy match constraints
#
                1(-1,1), 1(-1,1), 1(-1,1), 1(-1,1),
#
# the template proper
#
               (
                 150,  100,
                 144,  100,
                 138,  100,
                 132,  100,
                 127,  100,
                 122,  100,
                 118,  100,
                 113,  100,
                 108,  100,
                 103,  100,
                 101,  100,
                  99,  100,
                  97,  100,
                  94,  100,
                  92,  100,
                  90,  100,
                  88,  100,
                  86,  100,
                  84,  100,
                              ) ];
```

3. *Grammar of single synthetic template 1.AV.*

```
#
     Sigma = Template;
#
     Template = [
#
# dummy match constraints
#
                    1(-1,1), 1(-1,1), 1(-1,1), 1(-1,1),
#
# the template proper
#
                   (
                     150,  100,
                     144,  100,
                     138,  100,
                     132,  100,
                     127,  100,
                     122,  100,
                     118,  100,
                     113,  100,
                     108,  100,
                     103,  100,
                     101,  100,
                      99,  100,
                      97,  100,
                      94,  100,
                      92,  100,
                      90,  100,
                      88,  100,
                      86,  100,
                      84,  100,
                                ) ];
```

4. *Grammar for connected utterance example*

```
#
# Head/nucleus description
# D.L. 1984    file C.SYN
#
 Sigma = Head + Nucleus;
#
# phonetics of head unknown,
# so use reproduction of a PFVS head
#
 Head = THead;
#
# allow SDA to select from one of five tones.  Use the training
# templates of table 6.2, r = 2.
#
 Nucleus = TTone1, TTone2, TTone3,
          TTone4, TTone5;
#
# define head template as actual speech PFVS of similar length
#
 THead = [ 10(-1,1), 10(1,1), 10(-1,1), 10(1,1),

             (
               151,     4,
               250,    83,
               125,    86,
                95,    87,
               108,    78,
               101,    79,
               106,    80,
               107,    83,
               109,    84,
               108,    84,
               111,    83,
               123,    81,
               125,    78,
               129,    75,
               133,    75,
               140,    72,
               142,    72,
               147,    68,
               153,    64,
               153,    60,
               153,    52,
               131,    47,
                 0,     0,
                 0,     0,
                 0,     0,
                71,     6,
                87,     5,
                87,     4,
                66,     3,
               208,     3,
               227,     3,
                88,     5,
               109,     7,
               109,    60,
                82,    82,
               175,    90,
               144,    90,
                 144,    70,
                 142,    70,
                 142,    70,
                 140,    64,
                 135,    64,
                 140,    76,
                 196,    73,
                  54,    67,
                 123,    31,
                  53,    12,
                  94,    27,
                 128,    28,
                  85,    44,
                  83,    48,
                 101,    78,
                 158,    78,
                 126,    71,
                 126,    45,
                  90,    31,
                  93,     7,
                   0,     0
                ) ];
#
 TTone1 = [ 10(-1,1), 10(1,1), 10(-1,1), 10(1,1),
#
                (
                 153,    66,
                 120,    69,
                 117,    72,
                 109,    78,
                 112,    83,
                 111,    87,
                 108,    87,
                 112,    89,
                 113,    92,
                 112,    96,
                 112,    96,
                 112,    93,
                 111,    96,
                 113,   100,
                 109,    97,
                 111,    94,
                 108,    92,
                 107,    92,
                 107,    92,
                 105,    92,
                 104,    89,
                 104,    85,
                 101,    78,
                 100,    71,
                  96,    66,
                  94,    65,
                  96,    65,
                  92,    70,
                  92,    72,
                  91,    75,
                  98,    75,
                  87,    74,
                  95,    74,
                  95,    62,
                 88,    62
   ) ];
#
 TTone2 = [ 10(-1,1), 10(1,1), 10(-1,1), 10(1,1),
#
               (
                161,    19,
                161,    24,
                270,    51,
                136,    67,
                126,    78,
                120,    78,
                120,    81,
                120,    83,
                120,    84,
                120,    87,
                120,    88,
                123,    91,
                125,    91,
                129,    93,
                133,   100,
                140,   100,
                149,    97,
                156,    97,
                175,    94,
                192,    93,
                212,    89,
                232,    83,
                250,    81,
                270,    74,
                 99,    42
                 ) ];

#
 TTone3 = [ 10(-1,1), 10(1,1), 10(-1,1), 10(1,1),
#
               (
                 121,    61,
                 119,    63,
                 119,    67,
                 120,    64,
                 117,    65,
                 117,    64,
                 119,    65,
                 117,    67,
                 119,    67,
                 117,    69,
                 117,    69,
                 119,    70,
                 117,    70,
                 119,    69,
                 117,    69,
                 116,    69,
                 117,    69,
                 119,    70,
                 117,    72,
                 117,    75,
                 119,    75,
                 119,    74,
                 119,    74,
                  119,    73,
                  117,    74,
                  120,    76,
                  117,    77,
                  119,    77,
                  117,    77,
                  120,    79,
                  119,    80,
                  120,    82,
                  119,    91,
                  119,   100,
                  117,   100,
                  114,    95
               ) ];
#
 TTone4 = [ 10(-1,1), 10(1,1), 10(-1,1), 10(1,1),
#
               (
                  163,    59,
                  163,    60,
                  166,    60,
                  163,    62,
                  166,    63,
                  166,    64,
                  161,    61,
                  161,    62,
                  158,    62,
                  153,    65,
                  144,    66,
                  142,    66,
                  136,    68,
                  131,    68,
                  128,    67,
                  125,    68,
                  121,    68,
                  119,    68,
                  120,    67,
                  120,    66,
                  121,    67,
                  121,    67,
                  126,    68,
                  129,    69,
                  131,    69,
                  135,    71,
                  135,    77,
                  136,    80,
                  138,    91,
                  138,   100,
                  136,   100
               ) ];
#
 TTone5 = [ 10(-1,1), 10(1,1), 10(-1,1), 10(1,1),
#
               (
                  128,    78,
                  131,    93,
                  135,    98,
                  136,   100,
                  140,   100,
                  147,    99,
                153,    93,
                161,    90,
                166,    89,
                172,    86,
                181,    83,
                192,    83,
                200,    88,
                212,    81,
                217,    93,
                222,    96,
                232,    88,
                222,    87,
                217,    89,
                200,    89,
                185,    88,
                166,    88,
                149,    78,
                138,    75,
                117,    74,
                107,    73,
                 97,    70,
                 92,    65,
                 86,    60,
                 89,    60,
                 66,    60
            } };
#
```

5. Connected utterance grammar listing from Rule Translator

```
0 [ 4]    1.   Sigma            = PreHead + Head + Nucleus;
0 [ 6]    2.   Head             = THead ;
1 [11]    3.   Nucleus          = TTone1 , TTone2 ,
1 [11]    3.                      TTone3 , TTone4 ,
1 [11]    3.                      TTone5 ;
2 [11]    4.   THead            =
2 [11]    4.          [
2 [11]    4.          transformation bound on x dimension    1
2 [11]    4.          transformation bound on y dimension    1
2 [11]    4.          penalty factor on DTW                  1
2 [11]    4.       58(
2 [11]    4.                151,      4,
2 [11]    4.                250,     83,
2 [11]    4.                125,     86,
2 [11]    4.                 95,     87,
2 [11]    4.                108,     78,
2 [11]    4.                101,     79,
2 [11]    4.                106,     80,
2 [11]    4.                107,     83,
2 [11]    4.                109,     84,
2 [11]    4.                108,     84,
2 [11]    4.                111,     83,
2 [11]    4.                123,     81,
2 [11]    4.                125,     78,
2 [11]    4.                129,     75,
2 [11]    4.                133,     75,
2 [11]    4.                140,     72,
2 [11]    4.                142,     72,
2 [11]    4.                147,     68,
2 [11]    4.                153,     64,
2 [11]    4.                153,     60,
2 [11]    4.                153,     52,
2 [11]    4.                131,     47,
2 [11]    4.                  0,      0,
2 [11]    4.                  0,      0,
2 [11]    4.                  0,      0,
2 [11]    4.                 71,      6,
2 [11]    4.                 87,      5,
2 [11]    4.                 87,      4,
2 [11]    4.                 66,      3,
2 [11]    4.                208,      3,
2 [11]    4.                227,      3,
2 [11]    4.                 88,      5,
2 [11]    4.                109,      7,
2 [11]    4.                109,     60,
2 [11]    4.                 82,     82,
2 [11]    4.                175,     90,
2 [11]    4.                144,     90,
2 [11]    4.                144,     70,
2 [11]    4.                142,     70,
2 [11]    4.                142,     70,
2 [11]    4.                140,     64,
2 [11]    4.                135,     64,
2 [11]    4.                140,     76,
2 [11]    4.                196,     73,
2 [11]    4.                 54,     67,
2 [11]    4.                123,     31,
2 [11]    4.                 53,     12,
2 [11]    4.                   94,      27,
2 [11]    4.                  128,      28,
2 [11]    4.                   85,      44,
2 [11]    4.                   83,      48,
2 [11]    4.                  101,      78,
2 [11]    4.                  158,      78,
2 [11]    4.                  126,      71,
2 [11]    4.                  126,      45,
2 [11]    4.                   90,      31,
2 [11]    4.                   93,       7,
2 [11]    4.                    0,       0,
2 [11]    4.             )
2 [11]    4.           ] ;
2 [11]    5.    TTone1          =
2 [11]    5.           [
2 [11]    5.           transformation bound on x dimension    1
2 [11]    5.           transformation bound on y dimension    1
2 [11]    5.           penalty factor on DTW                  1
2 [11]    5.        35(
2 [11]    5.                  153,      66,
2 [11]    5.                  120,      69,
2 [11]    5.                  117,      72,
2 [11]    5.                  109,      78,
2 [11]    5.                  112,      83,
2 [11]    5.                  111,      87,
2 [11]    5.                  108,      87,
2 [11]    5.                  112,      89,
2 [11]    5.                  113,      92,
2 [11]    5.                  112,      96,
2 [11]    5.                  112,      96,
2 [11]    5.                  112,      93,
2 [11]    5.                  111,      96,
2 [11]    5.                  113,     100,
2 [11]    5.                  109,      97,
2 [11]    5.                  111,      94,
2 [11]    5.                  108,      92,
2 [11]    5.                  107,      92,
2 [11]    5.                  107,      92,
2 [11]    5.                  105,      92,
2 [11]    5.                  104,      89,
2 [11]    5.                  104,      85,
2 [11]    5.                  101,      78,
2 [11]    5.                  100,      71,
2 [11]    5.                   96,      66,
2 [11]    5.                   94,      65,
2 [11]    5.                   96,      65,
2 [11]    5.                   92,      70,
2 [11]    5.                   92,      72,
2 [11]    5.                   91,      75,
2 [11]    5.                   98,      75,
2 [11]    5.                   87,      74,
2 [11]    5.                   95,      74,
2 [11]    5.                   95,      62,
2 [11]    5.                   88,      62,
2 [11]    5.             )
2 [11]    5.           ] ;
3 [11]    6.    TTone2          =
3 [11]    6.           [
3 [11]    6.           transformation bound on x dimension    1
3 [11]    6.           transformation bound on y dimension    1
```

```
3 [11]    6.          penalty factor on DTW                    1
3 [11]    6.       25(
3 [11]    6.                161,      19,
3 [11]    6.                161,      24,
3 [11]    6.                270,      51,
3 [11]    6.                136,      67,
3 [11]    6.                126,      78,
3 [11]    6.                120,      78,
3 [11]    6.                120,      81,
3 [11]    6.                120,      83,
3 [11]    6.                120,      84,
3 [11]    6.                120,      87,
3 [11]    6.                120,      88,
3 [11]    6.                123,      91,
3 [11]    6.                125,      91,
3 [11]    6.                129,      93,
3 [11]    6.                133,     100,
3 [11]    6.                140,     100,
3 [11]    6.                149,      97,
3 [11]    6.                156,      97,
3 [11]    6.                175,      94,
3 [11]    6.                192,      93,
3 [11]    6.                212,      89,
3 [11]    6.                232,      83,
3 [11]    6.                250,      81,
3 [11]    6.                270,      74,
3 [11]    6.                 99,      42,
3 [11]    6.            )
3 [11]    6.          ] ;
3 [11]    7.    TTone3        =
3 [11]    7.          [
3 [11]    7.          transformation bound on x dimension    1
3 [11]    7.          transformation bound on y dimension    1
3 [11]    7.          penalty factor on DTW                  1
3 [11]    7.       36(
3 [11]    7.                121,      61,
3 [11]    7.                119,      63,
3 [11]    7.                119,      67,
3 [11]    7.                120,      64,
3 [11]    7.                117,      65,
3 [11]    7.                117,      64,
3 [11]    7.                119,      65,
3 [11]    7.                117,      67,
3 [11]    7.                119,      67,
3 [11]    7.                117,      69,
3 [11]    7.                117,      69,
3 [11]    7.                119,      70,
3 [11]    7.                117,      70,
3 [11]    7.                119,      69,
3 [11]    7.                117,      69,
3 [11]    7.                116,      69,
3 [11]    7.                117,      69,
3 [11]    7.                119,      70,
3 [11]    7.                117,      72,
3 [11]    7.                117,      75,
3 [11]    7.                119,      75,
3 [11]    7.                119,      74,
3 [11]    7.                119,      74,
3 [11]    7.                119,      73,
3 [11]    7.                117,      74,
3 [11]   7.                 120,     76,
3 [11]   7.                 117,     77,
3 [11]   7.                 119,     77,
3 [11]   7.                 117,     77,
3 [11]   7.                 120,     79,
3 [11]   7.                 119,     80,
3 [11]   7.                 120,     82,
3 [11]   7.                 119,     91,
3 [11]   7.                 119,    100,
3 [11]   7.                 117,    100,
3 [11]   7.                 114,     95,
3 [11]   7.            )
3 [11]   7.           ] ;
4 [11]   8.     TTone4          =
4 [11]   8.           [
4 [11]   8.           transformation bound on x dimension    1
4 [11]   8.           transformation bound on y dimension    1
4 [11]   8.           penalty factor on DTW                  1
4 [11]   8.        31(
4 [11]   8.                 163,     59,
4 [11]   8.                 163,     60,
4 [11]   8.                 166,     60,
4 [11]   8.                 163,     62,
4 [11]   8.                 166,     63,
4 [11]   8.                 166,     64,
4 [11]   8.                 161,     61,
4 [11]   8.                 161,     62,
4 [11]   8.                 158,     62,
4 [11]   8.                 153,     65,
4 [11]   8.                 144,     66,
4 [11]   8.                 142,     66,
4 [11]   8.                 136,     68,
4 [11]   8.                 131,     68,
4 [11]   8.                 128,     67,
4 [11]   8.                 125,     68,
4 [11]   8.                 121,     68,
4 [11]   8.                 119,     68,
4 [11]   8.                 120,     67,
4 [11]   8.                 120,     66,
4 [11]   8.                 121,     67,
4 [11]   8.                 121,     67,
4 [11]   8.                 126,     68,
4 [11]   8.                 129,     69,
4 [11]   8.                 131,     69,
4 [11]   8.                 135,     71,
4 [11]   8.                 135,     77,
4 [11]   8.                 136,     80,
4 [11]   8.                 138,     91,
4 [11]   8.                 138,    100,
4 [11]   8.                 136,    100,
4 [11]   8.            )
4 [11]   8.           ] ;
4 [11]   9.     TTone5          =
4 [11]   9.           [
4 [11]   9.           transformation bound on x dimension    1
4 [11]   9.           transformation bound on y dimension    1
4 [11]   9.           penalty factor on DTW                  1
4 [11]   9.        31(
4 [11]   9.                 128,     78,
4 [11]   9.                 131,     93,
4 [11]    9.                 135,      98,
4 [11]    9.                 136,     100,
4 [11]    9.                 140,     100,
4 [11]    9.                 147,      99,
4 [11]    9.                 153,      93,
4 [11]    9.                 161,      90,
4 [11]    9.                 166,      89,
4 [11]    9.                 172,      86,
4 [11]    9.                 181,      83,
4 [11]    9.                 192,      83,
4 [11]    9.                 200,      88,
4 [11]    9.                 212,      81,
4 [11]    9.                 217,      93,
4 [11]    9.                 222,      96,
4 [11]    9.                 232,      88,
4 [11]    9.                 222,      87,
4 [11]    9.                 217,      89,
4 [11]    9.                 200,      89,
4 [11]    9.                 185,      88,
4 [11]    9.                 166,      88,
4 [11]    9.                 149,      78,
4 [11]    9.                 138,      75,
4 [11]    9.                 117,      74,
4 [11]    9.                 107,      73,
4 [11]    9.                  97,      70,
4 [11]    9.                  92,      65,
4 [11]    9.                  86,      60,
4 [11]    9.                  89,      60,
4 [11]    9.                  66,      60,
4 [11]    9.            )
4 [11]    9.          ] ;
```

Appendix 4 Pascal procedure SRI - rule translation

The procedure SRI (SDA Rule Interpreter) translates the phonological rules and creates the intermediate representation. The source code for selected routines is given here to expand on the discussion of section 4.2, where a description of each routine is given.

```
procedure Sri;

  const

    (* lexical analysis *)

    Endsym = ';';
    Becomesym = '=';
    Plusym = '+';
    Minusym = '-';
    Cotempsym = '@';
    Pmetocsym = '@';
    Parasym = ',';
    Stopsym = ';';
    Termsym = '[';
    Mretsym = ']';
    Dotsym = '.';
    Lparsym = '(';
    Rparsym = ')';

    (* compilation control *)

    Csmode = False;
    EofVal = 32B;

  type
    CharSet = set of Char;

  var

    (* lexical analysis *)

    All: CharSet;
    Numerics: CharSet;
    Controls: CharSet;
    Letters: CharSet;
    Digits: CharSet;
    Others: CharSet;
    CharBuffer: Char;

    (* compilation control *)

    Number: Integer;
    Indch: MnChr..MChr;
    Remember: ANode;

    (* code fragments *)

    CodeFragment: ANode;
    TempA: AArea;

    (* link control *)

    RootArea: AArea;
    LinkFail: Integer;
    LinkCount: Integer;
    Found: Boolean;
```

```
procedure TranslateRule(var SyntaxF: Text;
                        var CodeAddress: ANode);

  var

    (* temporary addresses:
       general, csplus, csminus & cs *)

    Tna: ANode;
    I: ANode;
    Tcsp: ANode;
    Tcsm: ANode;
    An: AArea;
    Cn: CArea;

    (* code fragments *)

    CodeId: String;
    ACode: AArea;
    BCode: BArea;
    CCode: CArea;
    CodeCsPlus: ANode;
    CodeCsMinus: ANode;
    CodeIsTactic: Boolean;
```

```
procedure GetNextChar;

  const
    CommentSymbol = '';


  begin
    if Eof(SyntaxF) then CharBuffer := Chr(EofVal)
    else
      begin
      if Eoln(SyntaxF) then
        begin
        Readln(SyntaxF);
        GetNextChar;
        end
      else
        begin
        Read(SyntaxF, CharBuffer);
        if (CharBuffer = CommentSymbol) then
          begin
          Readln(SyntaxF);
          GetNextChar;
          end
        else (* buffer contains legal text *)
        end
      end

  end (*GetNextChar*) ;
```

```
procedure Skip(Cs: CharSet);


  begin
    while CharBuffer in Cs do
      begin
      GetNextChar;
      end;
  end (*Skip*) ;


procedure SkipUntil(Cs: CharSet);

  var
    Temp: CharSet;


  begin
    Temp := [Chr(0)..Chr(255)] - Cs;
    Skip(Temp)
  end (*SkipUntil*) ;


procedure GetStr(var S: String);

  var
    Diglets: CharSet;


  begin
    S := NilString;
    while not (CharBuffer in Letters) do GetNextChar;
    while (CharBuffer in (['0'..'9'] + Letters)) and
          (S.Len < Malfa) do
      begin
      S.Len := Succ(S.Len);
      S.Ch[S.Len] := CharBuffer;
      GetNextChar;
      end;
  end (*GetStr*) ;
```

```
procedure CheckSym(C: Char);


  begin
    Skip([Blank]);
    if CharBuffer <> C then
      begin
      Writeln('Error: expected=', C, ' found=', CharBuffer);
      Stop('Lex. Scan Fail.')
      end;
  end (*Sym*) ;


function Symeq(C: Char): Boolean;


  begin
    if CharBuffer = C then
      begin
      GetNextChar;
      Symeq := True
      end
    else Symeq := False
  end (*Symeq*) ;


procedure Sym(C: Char);


  begin
    Skip([Blank]);
    if not Symeq(C) then
      begin
      Writeln('Input contains "', CharBuffer, '",
               expected "', C, '".');
      Stop('Sym');
      end
  end (*Sym*) ;
```

```
procedure GetInt(var R: Integer);

  var
    I: Integer;
    Neg: Boolean;


  begin

    Skip([Blank]);
    I := 0;
    Neg := Symeq('+');
    Neg := Symeq('-');
    while (CharBuffer in ['0'..'9']) do
      begin
      I := I * 10 + Ord(CharBuffer) - Ord('0');
      GetNextChar;
      end;
    if Neg then R := - I
    else R := I;

  end;
```

```
procedure Term;

  const
    Dterm = True;

  var
    Tempi: Integer;
    I: RTra;
    J: 0..MSeg;
    R: Real;


  begin
    GetNextChar;
    with CCode do
      begin
      for I := Tra1 to Tra4 do
        begin (* load up transform info *)
        SkipUntil(['0'..'9', Parasym]);
        if CharBuffer = Parasym then
          begin
          (* load up default transform *)
          end
        else
          begin
          GetInt(Iteration[I]);
          Skip([Blank]);
          Sym(Lparsym);
          Skip([Blank]);
          GetInt(LwbTrans[I]);
          Skip([Blank]);
          Sym(Parasym);
          Skip([Blank]);
          GetInt(UpbTrans[I]);
          Skip([Blank]);
          Sym(Rparsym);
          Skip([Blank]);
          Sym(Parasym);
          end;
        end (*with*) ;
```

```
procedure GetInt(var R: Integer);

  var
    I: Integer;
    Neg: Boolean;


  begin

    Skip([Blank]);
    I := 0;
    Neg := Symeq('+');
    Neg := Symeq('-');
    while (CharBuffer in ['0'..'9']) do
      begin
      I := I * 10 + Ord(CharBuffer) - Ord('0');
      GetNextChar;
      end;
    if Neg then R := - I
    else R := I;

  end;
```

```
procedure Term;

  const
    Dterm = True;

  var
    Tempi: Integer;
    I: RTra;
    J: 0..MSeg;
    R: Real;


  begin
    GetNextChar;
    with CCode do
      begin
      for I := Tra1 to Tra4 do
        begin (* load up transform info *)
        SkipUntil(['0'..'9', Parasym]);
        if CharBuffer = Parasym then
          begin
          (* load up default transform *)
          end
        else
          begin
          GetInt(Iteration[I]);
          Skip([Blank]);
          Sym(Lparsym);
          Skip([Blank]);
          GetInt(LwbTrans[I]);
          Skip([Blank]);
          Sym(Parasym);
          Skip([Blank]);
          GetInt(UpbTrans[I]);
          Skip([Blank]);
          Sym(Rparsym);
          Skip([Blank]);
          Sym(Parasym);
          end;
        end (*with*) ;
        (* get template *)

        Skip([Blank]);
        Sym(Lparsym);
        J := 0;
        repeat
          J := Succ(J);
          GetInt(Template.V[J].F0);
          Skip([Blank]);
          Sym(Parasym);
          GetInt(Template.V[J].Cf0);
          Skip([Blank]);
        until (J = MSeg) or not Symeq(Parasym);
        Template.Leng := J;
        for J := MSeg downto Succ(Template.Leng) do
          Template.V[J] := NilPfvt;
        Sym(Rparsym);
        if Template.Leng > MSeg then
          Stop('Template too long in TERM.');
        if Template.Leng < 6 then
          Stop('Template too short in TERM.');

        (* exit from terminal node description *)

        Skip([Blank]);
        Sym(Mretsym);

        end;

    end (*Term*) ;
    (* get template *)

    Skip([Blank]);
    Sym(Lparsym);
    J := 0;
    repeat
      J := Succ(J);
      GetInt(Template.V[J].F0);
      Skip([Blank]);
      Sym(Parasym);
      GetInt(Template.V[J].Cf0);
      Skip([Blank]);
    until (J = MSeg) or not Symeq(Parasym);
    Template.Leng := J;
    for J := MSeg downto Succ(Template.Leng) do
      Template.V[J] := NilPfvt;
    Sym(Rparsym);
    if Template.Leng > MSeg then
      Stop('Template too long in TERM.');
    if Template.Leng < 6 then
      Stop('Template too short in TERM.');

    (* exit from terminal node description *)

    Skip([Blank]);
    Sym(Mretsym);

    end;

end (*Term*) ;
```

```
procedure Seq;

  var
    TempAArea: AArea;
    TempDep: ANode;


  begin
    Skip([Blank]);
    case CharBuffer of

      Stopsym:

      Termsym: Term;

      Plusym:

      (* handle type information first *)

        begin
        ACode.B.Sequ := True;
        ACode.B.Syntag := False;
        GetNextChar;
        Seq
        end;

      Parasym:

      (* handle symbol information *)

        begin
        ACode.B.Sequ := False;
        ACode.B.Syntag := True;
        GetNextChar;
        Seq
        end;

      otherwise

        (* non-terminal processing, process dependents*)

        if UpbRow(ACode.P) < MRow then
          begin
          NewA(ACode.P[UpbRow(ACode.P) + 1]);
          TempAArea := NilAArea;
          GetStr(TempAArea.Id);
          PutA(ACode.P[UpbRow(ACode.P)], TempAArea);
          Seq (* go on to next term *)
          end
        else Stop('To many RH terms for SEQ.')

      end (*case*) ;

  end (*Seq*) ;
```

```
procedure Rh;

  var
    Temp: CharSet;


  begin (*rh*)
    Skip([Blank]);
    case CharBuffer of

      Termsym:
        begin
        CodeIsTactic := False;
        Term;
        end;

      Cotempsym:
        begin
        CodeIsTactic := True;
        Seq
        end;

      otherwise
        begin
        CodeIsTactic := True;
        Seq
        end

      end (*case*)

  end (*Rh*) ;


function Cs(C: Char): ANode;

  var
    Csptr: ANode;
    An: AArea;


  begin
    Skip([Blank]);
    if Symeq(C) then
      begin
      NewA(Csptr);
      An := NilAArea;
      GetStr(An.Id);
      PutA(Csptr, An);
      Cs := Csptr
      end
    else Cs := NilA
  end (*Cs*) ;
```

```
procedure Lh;


  begin
    GetStr(CodeId);
    CodeCsPlus := Cs(Plusym);
    CodeCsMinus := Cs(Minusym);
  end (*Lh*) ;


begin (*TranslateRule*)

  CodeId.Len := 0;
  ACode := NilAArea;
  BCode := NilBArea;

  (* parse syntax rule *)

  Lh;
  Sym(Becomesym);
  Rh;
  CheckSym(Endsym);
  SkipUntil((Digits + Letters) + [Chr(EofVal)]);

  (* assemble network code *)

  An := ACode;
  An.Id := CodeId;
  if CodeIsTactic then
    begin
    An.B.Sort := Tactic;
    An.C := NilC;
    end
  else
    begin
    An.B.Sort := Terminal;
    NewC(An.C);
    PutC(An.C, CCode);
    end;
  NewA(CodeAddress);
  PutA(CodeAddress, An);

end (*TranslateRule*) ;
```

```
procedure Link(var Goal: ANode;
               CodeAddress: ANode;
               var Found: Boolean;
               var Plus: Integer);

  var
    An, An2, Code: AArea;
    I, J: RRow;
    TempDep: ANode;


  begin

    if OkA(Goal) and OkA(CodeAddress) then
      begin
      GetA(Goal, An);
      GetA(CodeAddress, Code);

      if An.Id.Ch = Code.Id.Ch then

      (* link this node here *)

        begin
        Plus := Plus + 1;
        Found := True;

        (* decide if terminal node *)

        if Code.B.Sort = Tactic then
          begin
          An.B.Sort := Tactic;
          An.B := Code.B;
          An.C := NilC;
          for J := 1 to UpbRow(Code.P) do
            begin
            NewA(An.P[J]);
            GetA(Code.P[J], An2);
            PutA(An.P[J], An2);
            end;
          end
        else
          begin
          An.B.Sort := Terminal;
          An.C := Code.C;
          end (*IF code.b.sort = tactic *)

        end (*IF an.id.ch = code.id.ch*)

      else
        begin
        for I := 1 to UpbRow(An.P) do
          Link(An.P[I], CodeAddress, Found, Plus);
        end;

      PutA(Goal, An);
      end (*ok clause*) ;

  end (*Link*) ;
```

```
begin (*Sri*)

  SetA(Into);
  SetC(Into);
  InitSri;

  (* translate first rule of grammar *)

  Number := 1;
  TranslateRule(SyntaxF, Sys.Root);
  WrCode(Lf, Number, Sys.Root);

  (* go on translating rules until end of grammar *)

  while not Eof(SyntaxF) do
    begin

    (* translate *)

    TranslateRule(SyntaxF, CodeFragment);
    Number := Succ(Number);
    WrCode(Lf, Number, CodeFragment);

    (* link CODE FRAGMENT to tree of SYS.ROOT *)

    Found := False;
    LinkCount := 1;
    LinkFail := 0;
    Link(Sys.Root, CodeFragment, Found, LinkCount);

    (* release CODE FRAGMENT's space,
       any C AREAs unaffected        *)

    JunkA(CodeFragment);

    (* diagnostics *)

    if not Found then LinkFail := 1;

    end (*do,translation*) ;

  (* finish up *)

  Writeln(Lf);
  Close(SyntaxF);

end (*Sri*) ;
```

Appendix 5 Pascal analysis routines

The program ANALYSIS performs the syntax directed analysis of the PFVS. The source code for selected routines is given here to expand on the discussion of section 4.2, where a brief description of each routine is given.

```
(* program Analysis; *)

procedure Tsh(Pat: CNode (*pointer to pattern*) ;
              var DataLink: Integer (*start of PFVS*) ;
              var Error: Real (*returned DP score*) ;
              Print: Boolean (*plot listing flag*)
              );

  const
    Range = 5 (*range of DP endpoints*) ;
    R = 3 (*adjustment window length*) ;
    Big = vbig;
    Iformat = 5;

  var
    PArea: CArea;
    X, Weight: Real;
    I: - Range..Range;
    Score: array [ - Range..Range] of Real;
    Newleng, OldLeng, K, J, UpbWarp: Integer;
```

```
procedure Dp(Msp: Integer (*match start*) ;
            Nsp: Integer (*match end*) ;
            Ref: Segtype (*pattern*) ;
            var Score: Real (*dp match result*) ;
            Print: Boolean (*warp information*)
            );

  const
    Gbig = vbig;
    R1 = 8 (*r + 2*) ;

  var
    G: array [ - R1..R1, 0..MSeg] of Real
       (*virtual array of TW paths*) ;
    W: - R1..R1 (*index into virtual array*) ;
    X, Y: 0..MSeg (*indices into virtual array*) ;
    Z: array [0..2] of Real (*temp. results*) ;
    Tnd: Real (*time normalised score*) ;
    Dist: Real (*temp. distance*) ;
    M: Integer (*temp. minpos*) ;

  (*frame-to-frame distance function*)

  function D(X, Y: Integer): Real;

  (*mapping function into virtual array*)

  function F(X, Y: Integer): Integer;
```

```
begin

  (*initialise viritual array*)

  for W := - R1 to R1 do
    for Y := 0 to MSeg do G[W, Y] := Gbig;

  (*calculate all local optimal transitions
    within adjustment window              *)

  for Y := 1 to Ref.Leng do
    begin
    for X := 1 to Nsp do
      begin
      Dist := D(X, Y);
      if (X = 1) and (Y = 1) then G[F(X, Y), Y] := Dist
      else
        begin
        if (F(X, Y) >= - R) and (F(X, Y) <= R) then
          begin
          Z[0] := G[F(X, Y - 1), Y - 1] + Dist;
          Z[1] := G[F(X - 1, Y - 1), Y - 1] + 2 * Dist;
          Z[2] := G[F(X - 1, Y), Y] + Dist;
          G[F(X, Y), Y] := Min(Z);
          end (*if,within adjustment window*)
        end (*if,not initial calculation*) ;
      if (X = Nsp) and (Y = Ref.Leng)
      then Tnd := G[F(X, Y), Y]
      end (*for,x*) ;
    end (*for,y*) ;
  Tnd := Tnd / (Nsp + Ref.Leng);
  Score := Tnd;

end (* Dp *) ;
```

```
begin

  (* get the pattern template pointed to by PAT *)

  GetC(Pat, PArea);

  (* DP match for range of endpoints *)

  for I := - Range to Range do Score[I] := Big;
  for I := - Range to Range do
    begin
    Dp(DataLink, PArea.Template.Leng + I, PArea.Template,
       Score[I], False);
    end;

  (* calculate minimum for range *)

  J := Minpos(Score);
  X := Min(Score);
  Newleng := PArea.Template.Leng + J;
  Dp(DataLink, Newleng, PArea.Template, X, Print);
  OldLeng := DataLink;
  DataLink := DataLink + Newleng;
  Error := Error + X;

end (* Tsh *) ;
```

```
procedure TraceMatch(Goal: ANode;
                     var T: Text);

  var
    An: AArea;
    TempDep: RRow;
    I: RSeg;


  begin
    GetA(Goal, An);
    case An.B.Sort of
      Tactic:
        for TempDep := 1 to UpbRow(An.P) do
          if OkA(An.P[TempDep]) then
TraceMatch(An.P[TempDep], T);
      Terminal: Writeln(T, An.Id.Ch);
      end (*case,sort*) ;
  end (* TraceMatch *) ;
```

```
procedure Parse(Goal, Trace: ANode;
                var DataLink: Integer;
                var Sum: Real);

  var

    DepMin: ANode;
    ArgMin, ArgTemp, ArgTemp2: RRow;
    DataLinkMin: Integer;
    SumMin: Real;
    An, AnTemp: AArea;
    Val: array [1..MRow] of Real;
    Link: array [1..MRow] of Integer;


  begin

    GetA(Goal, An);
    An.B.State := Blocked;
    case An.B.Sort of

      Terminal: (* apply template matching to input sequence *)

        begin
        Tsh(An.C, DataLink, Sum, False);
        end (*terminal*) ;
```

```
Tactic: (* continue parsing down tree *)

  begin

  case An.B.Sequ of

    True:

     (* take all deps into account: returned values are the
       incremented values of sum error & current DataLink *)


      begin
      for ArgTemp := 1 to UpbRow(An.P) do
        begin
        Parse(An.P[ArgTemp], Trace, DataLink, Sum);
        end;
      end;

    False: (*go through all choices of dependent - choose
              best*)

      begin

      for ArgTemp := 1 to MRow do Val[ArgTemp] := Vbig;
      for ArgTemp := 1 to UpbRow(An.P) do
        begin
        Link[ArgTemp] := DataLink;
        Val[ArgTemp] := Sum;
        Parse(An.P[ArgTemp], Trace, Link[ArgTemp],
              Val[ArgTemp]);
        end (*for,ArgTemp*) ;

      SumMin := Min(Val);
      ArgMin := Minpos(Val);
      DataLinkMin := Link[ArgMin];
      DepMin := An.P[ArgMin];
```

```
          (* return minimising choice *)

          Sum := Sum + SumMin;
          DataLink := DataLinkMin;

          (* rejected recognition paths have been
            de-activated, activate the chosen path *)

          for ArgTemp := UpbRow(An.P) downto Succ(ArgMin)
          do
            begin
            ArgTemp2 := ArgTemp;
            JunkA(An.P[ArgTemp2]);
            end;
          for ArgTemp := Pred(ArgMin) downto 1 do
            begin
            ArgTemp2 := ArgTemp;
            JunkA(An.P[ArgTemp2]);
            end;

          end (*sequ=false*) ;

        end (*case,sequ*) ;

      end (*sort=tactic*) ;

    end (*case,an.b.sort*) ;

end (*parse*) ;
```

```
procedure Return(Goal, Trace: ANode;
                 var DataLink: Integer;
                 var Sum: Real);

  var

    DepMin: ANode;
    ArgMin, ArgTemp: RRow;
    DataLinkMin: Integer;
    SumMin: Real;
    An, AnTemp: AArea;
    Val: array [1..MRow] of Real;
    Link: array [1..MRow] of Integer;


  begin

    GetA(Goal, An);
    An.B.State := Blocked;
    case An.B.Sort of

      Terminal: (* apply template matching to input sequence *)

        begin
        Tsh(An.C, DataLink, Sum, True);
        end (*terminal*) ;
      Tactic: (* continue parsing down tree *)

        begin
        for ArgTemp := 1 to UpbRow[An,P] do
          if OkA(An,P[ArgTemp]) then
            begin
            Return(An,P[ArgTemp], Trace, DataLink, Sum);
            end;
        end (*serl+tactic*) ;

      end (*case,an,h,serl*) ;

  end (*return*) ;

begin

  (* initialise *)

  OpenChannels;
  UsageInfo(Sys, Outof);
  Set4 (Outof);
  [illegible] (Outof).
  LoadPfxs;

  (* parse [illegible] sequence *)

  DataLink := 0 ;
  Sum := 0,0;
  Parse(Sys,Root, Trace, DataLink, Error);
  DataLink := 1;
  Sum := 0,0;
  Return(Sys,Root, Trace, DataLink, Error);
  DataLink := [illegible];
  (* identify session *)

  Writeln(Tf);
  Writeln(Tf, 'Syntax Directed Prosodic Analysis');
  Writeln(Tf);
  Writeln(Tf, 'intermediate code from ', NodeFld, '.A and .C');
  Writeln(Tf, 'speech data from        ', PfvsFld);
  Writeln(Tf, 'results to              ', NodeFld, '.RES');
  Writeln(Tf);

  (* give results *)

  Writeln(Tf, 'Analysis returns match error ', Error: 8,
            ' for ', DataLink / 100: 8: 2,
            ' seconds of input.');
  Writeln(Tf);

  (* trace best fitting sequence of matched templates *)

  Writeln(Tf,'The sequence of terminal symbols achieving',
           ' that score is' );
  Writeln(Tf);
  TraceMatch(Sys.Root, Tf);
  Writeln(Tf);
  Writeln(Tf);

  (* produce labelled bracketing of terminal symbols *)

  Writeln(Tf, 'The phrase structure as analysed',
            ' by the grammar is');
  Writeln(Tf);
  Writeln(Tf);
  LabelledBracketing(Sys.Root, Tf);
  Writeln(Tf);
  Writeln(Tf);

end (*analysis*) .
```