An Artificial Intelligence Language to Describe Extended Procedural Networks

Ettore Merlo

School of Computer Science

McGill University, Montreal

May 1989

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Artificial Intelligence Language to Describe Procedural Networks

To my family

TABLE OF CONTENTS

ABSTRACT

This thesis introduces a new language for Artificial Intelligence applications that allows to integrate symbolic and numerical computation. This language is applied to Automatic Speech Recognition (ASR).

Speaker-independence and large lexicon access are still two of the greatest problems in automatic speech recognition. Cognitive and information-theory approaches try to solve the recognition problem by proceeding in almost opposite directions. The former relies on knowledge representation, reasoning and perceptual analysis, while the latter is in general based on numerical algorithms.

Progress is expected from the integration of the two mentioned approaches. Artificial intelligence techniques are often used in the cognitive approach, but these techniques usually lack of sophisticated numerical support. The Extended Procedural Network constitutes a general AI framework which integrates powerful numerical strategies including stochastic techniques and symbolic computation.

The framework has been tested on difficult problems in speech recognition, including speaker-independent letter and digit recognition, speaker-independent vowel and diphthong recognition, and access to a large lexicon.

Various experiments and comparisons have been executed on a large number of speakers and the results are reported.

RESUME

Cette thèse presente un nouveau langage pour des applications en intelligence artificielle qui permettent l'intégration entre les calculs symbolique et numerique. Le langage présenté a été appliqué à la reconnaissance automatique de la parole (RAP).

L'indépendance du locuteur et l'accès à un grand lexique sont encore deux des plus importants problèmes en reconnaissance automatique de la parole. L'approche cognitive et celle de la théorie de l'information essaient de résoudre le problème en l'approchant de deux directions quasiment opposées. La première se base sur la représentation de la connaissance, le raisonnement et l'analyse de la perception, alors que la deuxième est en général basée sur des algorithmes numériques.

L'intégration des deux méthodes peut apporter des progrès. Les techniques d'intelligence artificielle sont souvent utilisées dans l'approche cognitive mais, actuellement, elles manquent de support numérique sophistiqué. Le réseau procédural etendu constitue une structure qui intègre de puissantes stratégies numériques incluant des techniques statistiques et de calcul symbolique.

Le modèle a été testé sur des problèmes difficiles en reconnaissance de la parole, incluant la reconnaissance indépendante du locuteur des lettres et des chiffres, des voyelles et des diphtongues, et l'accès à un grand lexique.

Des expeŕiences et des comparaisons différentes ont été faites sur un grand nombre de locuteurs et les résultats ont été rapportés.

ACKNOWLEDGEMENTS

I would like to thank Prof. Renato De Mori, my supervisor, for his remarkable guidance, encouragement and support. I want to express him my gratitude for providing the harmonious relationship, both from an academical and a personal point of view, that helped me in overcoming many difficulties.

I want to thank NSERC, "Consiglio Nazionale delle Ricerche" - Roma - Italy, and "Gruppo Dirigenti FIAT" - Torino - Italy, for their financial support that allowed me to undertake and conclude these researches and studies.

I am deeply indebted with the staff and management of the Centre de Recherche en Informatique de Montreal for their kindness in supplying valuable technical support together with a friendly and professional working environment.

I also want to thank the members of the Ph. D. committee Prof. N. Madhavji and Prof. M. Newborn for their advice.

Many fellow students and collegues helped me through the development on my thesis with valuable discussions and programs. In particular I want to thank Mathew Palakal, Jean Rouat, Regis Cardin, Yoshua Bengio, and Piero Cosi. I want to thank Robert Prager for the many suggestions that arose during the proofreading of the thesis.

My gratitude goes to the staff of the School of Computer Science at McGill University for their assistance and kindness. Thanks to Lorraine Harper for carefully typesetting part of this thesis.

I would like to thank my family for the support and encouragement they gave me at all levels during the many years of study.

I want to thank my girlfriend Christine and all my friends for the beautiful years I have spent in Quebec with them and for their moral support.

CHAPTER - 1

1 - INTRODUCTION

The design of the Extended Procedural Network (EPN) has its motivation in the need of integrating stochastic methods with symbolic computation. Generally speaking intelligent systems with numerical attachements can only deal with ad hoc numerical treatment. On the other side powerful numerical systems like those derived from pattern recognition cannot easily model non numerical strategies (like chaining rules, for example). The EPN model bridges the gap between these two philosophies. Any hybrid paradigm is easily described with the EPN model and the model itself encourages the study and the expression of hybrid strategies to solve problems that are hard to solve using a single paradigm. A discussion on the need of probability in A.I. systems can be found in [12]. Without taking such a radical approach and provided that the reasoning scheme can be represented by a network-like structure, efficient algorithms for probabilistic inference can be taken from the theory of Markov models and inserted into an A.I. environment. The details of the EPN model and the originality with respect to the existing network approaches to A.I. are given in this thesis.

The EPN model has been tested on several applications concerning Automatic Speech Recognition (ASR). Speaker-independent ASR by computers of large or difficult vocabularies by computers is still an unsolved task, especially if words are pronounced connectedly. Efforts and progress towards the solution of this problem are reported in the recent literature [8], [45] ,[49], [52], [53].

Recent results on ASR and Speech Analysis suggest that progress in designing recognition devices and in advancing speech science knowledge may arise from an integration of the so called **cognitive** and **information-theoretic** approaches [53].

The cognitive approach attempts to infer analytic knowledge about possible speech invariants and their relations. Work by Zue [93], Klatt [47], Stevens [83] and De Mori et al. [19], [24] are along this line.

The contribution proposed in this thesis is based on the following considerations. First, if large or difficult vocabularies have to be recognized when words are pronounced by many speakers, it is advisable to consider a (possibly small) set of **Speech Units** (SU) with which all the words and word concatenations can be represented by **compilation**. A relation between a word W and its SUs can be represented by a limited number of basic **prototypes** and a description of their distortions observed when W is pronounced by a large population of speakers in different contexts. Distortions introduce ambiguities in the relation $R_1(W,SU)$ between W and SUs. In order to make ambiguous relations more useful, for example, for recognition purposes, their statistics can be taken into account.

Second, the knowledge we have about production and perception of phonemes, diphones and syllables can be useful for conceiving **prototypes of Speech Units**. SU prototypes can be characterized by a **redundant set** of **Acoustic Properties (AP)**. A relation $R_2(SU,AP)$ between a Speech Unit and its APs is

ambiguous because acoustic properties can be distorted, missed or inserted in a particular instantiation of an SU. This is due to context, inter and intra-speaker variability. A **performance model** of such alterations can be built using statistical methods. This is called the Information-Theoretic approach.

The information theoretic approach is based on a performance model containing states and transitions between any pair of states [8]. Probabilities that the system is in any of the model states or is changing state through any of the allowed transitions can be learned. Furthermore, the model generates in each state or in each transition, observable system parameters or descriptors according to some statistical distribution.

Whether knowledge about speech analysis, synthesis and perception should be taken into account or not in ASR is still the subject of discussions among the researchers in the field. Investigating the possibility of using acoustic property descriptors for ASR is attractive. Nevertheless, an ASR system based on acoustic property descriptors is not very efficient if the set of properties used and the algorithms for their extraction are not well chosen and conceived. Notice that property descriptors **describe** the speech data and do not **interpret** them. Descriptors cannot be false or ambiguous, rather they can be **insufficient** or redundant for interpreting speech. For this reason it is important to start an investigation on property descriptors based on properties which are expected to be robust, speaker-independent cues of fundamental phonetic events. These properties and the algorithms that extract them may have different performance

and degrees of success in different cases. For this reason, a certain redundancy in the number of properties used for characterizing a phonetic event may be useful.

Remarkable work has been done so far on spectrogram reading [92]. A number of APs for SUs has been identified with such an effort. Attempts have also been made in order to extract some properties automatically and use them for ASR [21].

Knowledge about spectrograms is incomplete. We know that some properties that can be detected are relevant for perception. The same property may appear in slightly different patterns corresponding to different pronunciations of the same word because of inter and intra-speaker variations. It is important to characterize knowledge about such variations. This characterization has to be statistical because we do not have other types of knowledge on how basic word pattern prototypes are distorted when different speakers pronounce that same word. On the other hand, it is very important to characterize word prototypes in terms of properties that are relevant for speech production and for perception. **Property based prototypes** of words or SUs may describe a large variety of patterns not only because properties are distorted, but also because some properties are missed or some unexpected properties have been inserted. Insertions and deletions can be often characterized by deterministic rules reflecting basic coarticulation knowledge, but in many cases they cannot be fully explained and are better characterized by statistical methods.

Based on the above considerations, the application of the EPN to speech recognition represents an attempt to integrate **knowledge-based extraction** of relevant speech properties and **statistical modeling** of their distortions.

Furthermore, the choice of APs is such that the essential information for reconstructing understandable speech is preserved.

CHAPTER - 2

2 - EXTENDED PROCEDURAL NETWORKS

Chapter 2 describes a language based on Extended Procedural Networks (EPN). Section 2.1, 2.2, and 2.3 respectively present the formal model, the supervisor strategy and the related programming language. The remaining sections show how the EPN model can include a number of A.I. paradigms.

2.1 - THE MODEL

An Extended Procedural Network (*EPN*) can be described with a formalism similar to that used for an Augmented Transition Network Grammar (*ATNG*) [90]. This formalism has been successfully used for Natural Language and Pattern Recognition [32]. The novelty in the approach of the EPN consists in the tight integration between symbolic and numerical computation. The result is a new model which is able to describe and implements strategies that cannot be handled by ATNG's. Examples of some of the strategies that can be defined in the EPN model can be found in section 2.4. ATNG's and their applications are of course a special case of the much more powerful EPN model.

An EPN is a 5-tuple

$$EPN=\{j,Q,A,q_0,q_f\} \tag{2.1}$$

where j is the network identifier, Q is a finite set of states, A is a finite set of directed arcs, $q_o \in Q$ is the initial state and q_f is the final state. Without any loss

of generality only EPNs with a single initial state and a single final state are considered.

Each arc $a_i \in A$ is a 5 - tuple:

$$a_i=(q_{bi},q_{ei},P_i,condition_i,action_i) \tag{2.2}$$

where:

- $q_{bi} \in Q$ is the starting state of a_i
- $q_{ei} \in Q$ is the terminal state of a_i
- P_i is a measure associated to the arc (it can be a weight or a probability according to the scoring method used by the EPN supervisor described later on)
- $condition_i$ is a condition which has to be tested and is associated to the arc
- $action_i$ is an executable action associated to the arc

The conditions can be categorized in two classes:

COND n

refers to a user defined condition n.

DEFAULT r

refers to a default condition (it is satisfied only if no other transition of any arc whose starting state is q_{bi} returns a score greater than r).

The actions are executed by the EPN supervisor and can be categorized in eight classes:

EXE n

executes a user defined action; such an action is usually a "matcher" which performs some computations on the input data and returns a result.

PUSH i

is defined as follows. Let's assume that EPN_j has an arc that contains PUSH i. Let π_j be the process that executes EPN_j. When the arc is reached whose associated action is PUSH i, the execution of π_j is suspended. The state of π_j is pushed on the top of the stack of the EPN supervisor. A new process π_i that executes EPN_i is created and executed. When the final state of EPN_i is reached, the last arc of EPN_i is considered. It has associated either a POPABS f or a POPCOND f action. This action is executed. It returns scores computed by EPN_i. These scores are passed to π_j whose execution is resumed while π_i terminates.

POPABS f

is associated to the final state of an EPN. It stops the execution of the current network process as soon as t is executed. The result of the execution of the user defined function f is returned.

POPCOND f

It stops the execution of the current network if the final state is the only active state in the current column of the trellis, i. e. if all the actions associated to the paths in the network leading to the final state have been executed. If this condition is satisfied, then the result of the execution of the

user defined function f is returned.

JMP

makes the score associated to q_{bi} propagate to q_{ei} according to the measure P_i associated to the transition

INHIBIT a r

inhibits the activation of the arc a, if the score associated to q_{bi} is greater than r

EMPTY

propagates the score $f_t(P_i,\text{score}(q_{bi}))$ to q_{ei} (see the EPN supervisor section) in the same column of the trellis.

EXPAND i

the subnetwork EPN_i is substituted to the EXPAND transition in such a way that the initial state of the subnetwork coincides with the startpoint of the transition and the final state of the subnetwork coincides with the transition endpoint

PUSH and EXPAND implement a hierarchical strategy respectively model and data driven. The difference between the two actions can be better understood if we draw the corresponding trellis.

Figure 2.2 represents the trellis corresponding to the network of Figure 2.1 (PUSH) and Figure 2.4 corresponds to Figure 2.3 (EXPAND).

A set of built-in functions which modify the structure and the parameters of the

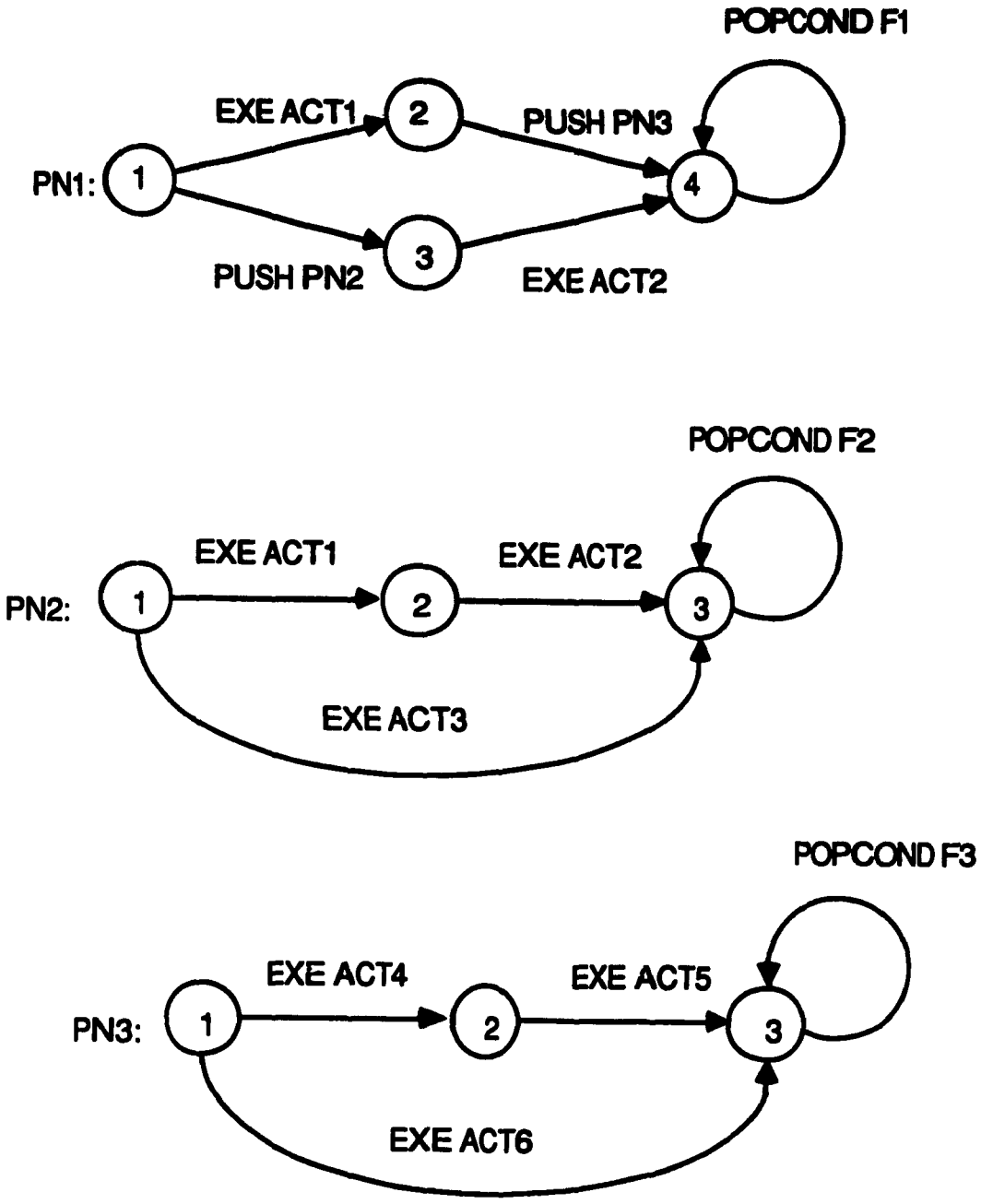


Figure 2.1: Example of EPN with PUSH actions

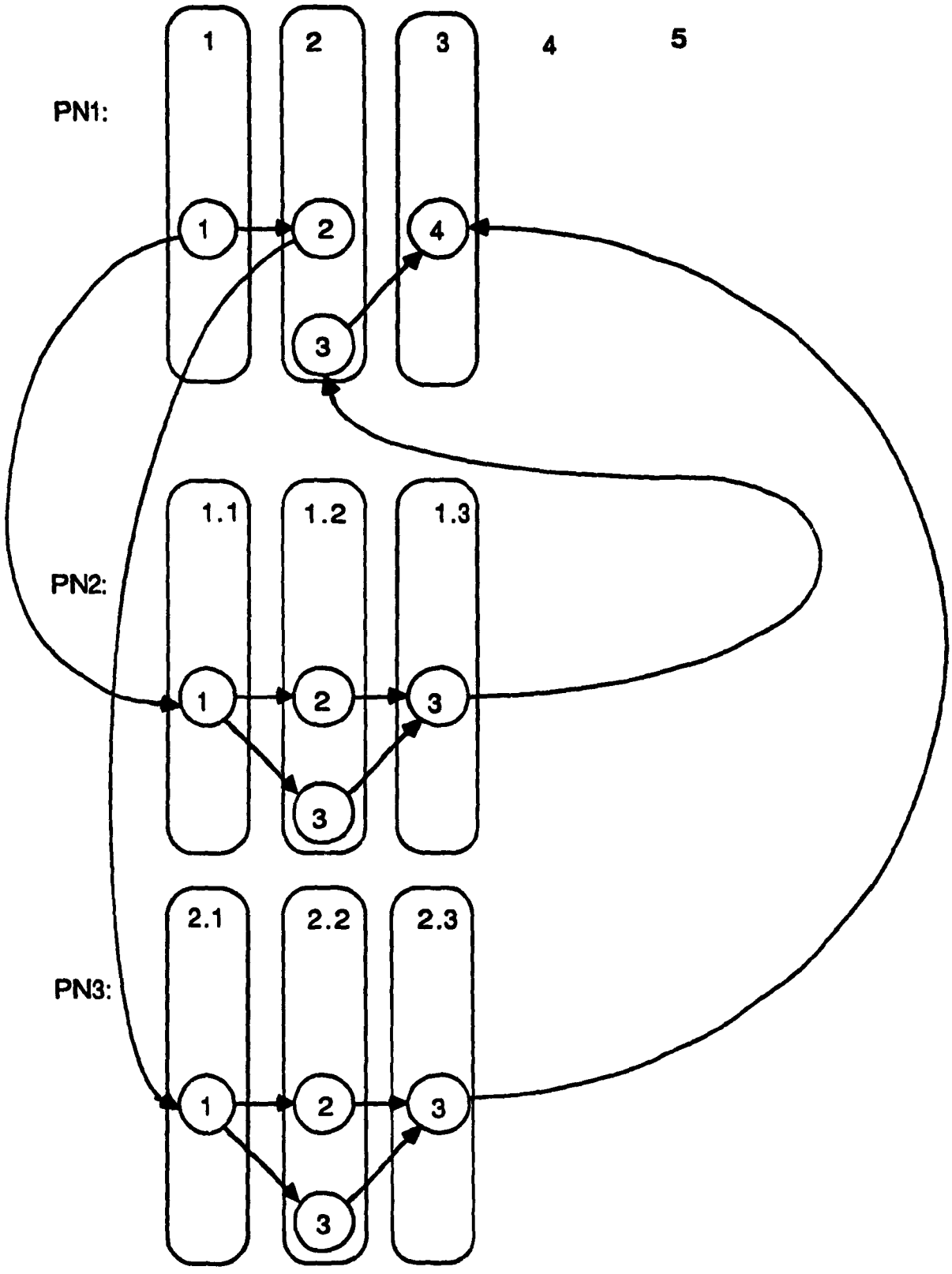


Figure 2.2: Trellis corresponding to the network of Figure 2.1

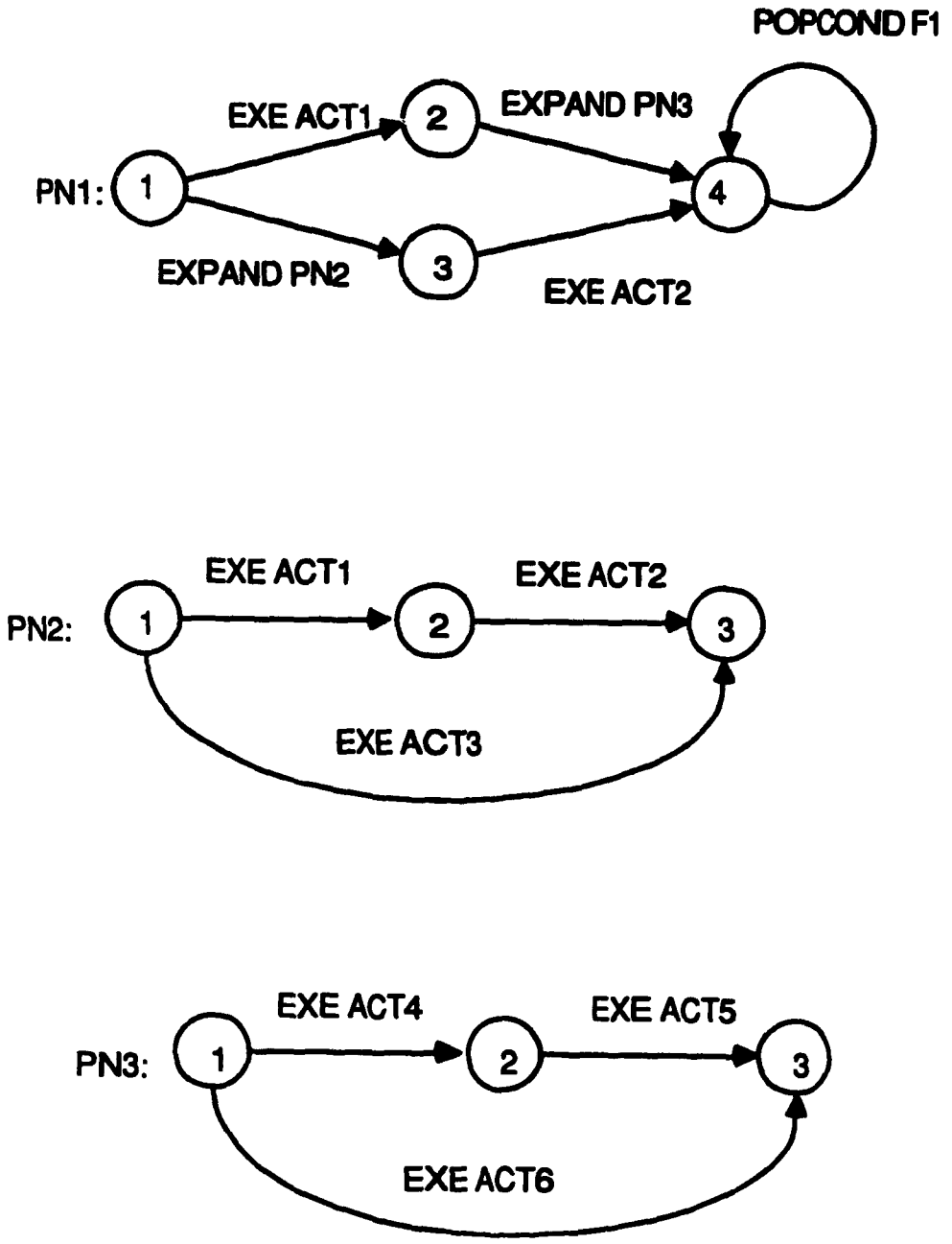


Figure 2.3: Example of EPN with EXPAND actions

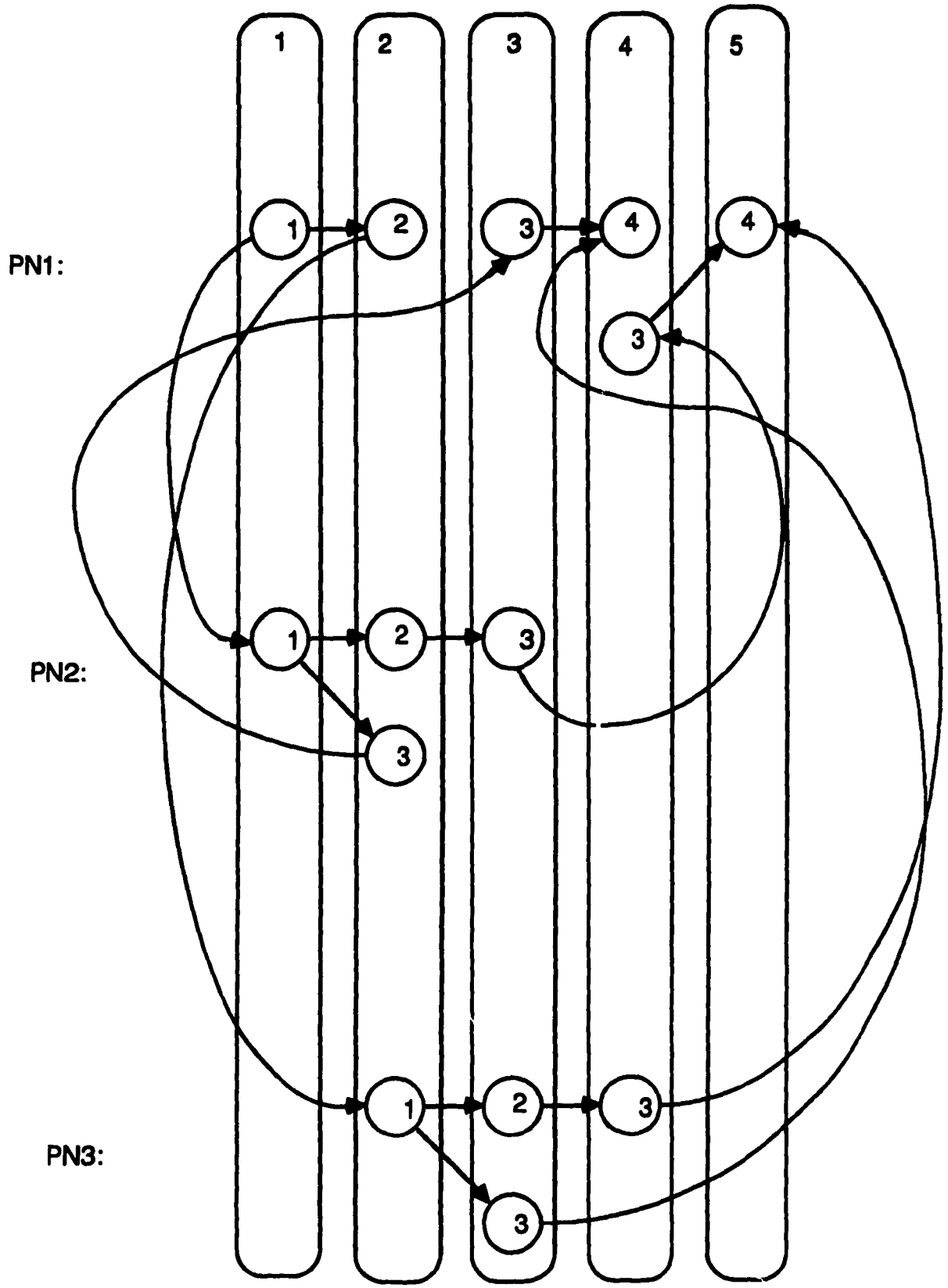


Figure 2.4: Trellis corresponding to the network of Figure 2.3

EPN is defined as follows:

CREATE transition trans_param

creates a new transition in the network whose transition parameters are specified as action parameters.

DELETE transition

deletes a transition

SET_PARAM transition trans_param

modifies the parameters of the specified transition

GET_PARAM transition param

returns in the variable param the parameters of the specified transition.

2.2 - THE SUPERVISOR

Several strategies can be applied in order to build a state space of hypotheses and to find the most plausible one. The supervisor considers two symmetrical strategies: forward strategy and backward strategy. Most of the applications do not require backward strategy.

Let a_i be an arc of an EPN. The contribution of such an arc is

$$u_i = f_i(p_i, g_i(cond_i), h_i(act_i)) \tag{2.3}$$

where:

- p_i is the score associated with a_i,
- g_i is a function which returns the evidence of satisfaction of $condition_i$,

- h_i is the function which returns the value computed by the $action_i$ and
- f_i is the function which combines the values of its arguments in order to give the contribution of the arc.

According to the definition of $p_i, g_i, h_i,$ and f_i, several interpretations of the contribution of an arc are possible. Sections 2.4 and 4.2.1 show how the mentioned parameters $p_i, g_i, h_i,$ and f_i, have to be characterized to specify various arc contributions in different paradigms.

The supervisor forward strategy calculates the compound contribution of all the transitions in a dynamic programming style. Partial computations (partial scores and contextual information) are stored in a state buffer associated with states in the trellis data structure. Every column of the trellis corresponds to a computational step in the dynamic programming process. Initially at computational time (i.e. discrete step) 0, only the initial state of the network belongs to the 0-th column. The associated score is supplied by the user as a parameter of the network and depends on the interpretation of the reasoning process. If a stochastic approach has been taken, i. e. measures are probabilities and the function f_i is the multiplicative function, then it is reasonable to initialize the score of the initial state to 1. This interpretation leads us to understand the scores as the probability of having reached a particular state of the network at a particular computational step. If the network is used to evaluate a distance then, for example, the measures are costs, the function f_i is an additive function and

the score of the initial state at step 0 would be reasonably set to 0.

The column 0 of the trellis would be subsequently completed by the states and contributions which arise from EMPTY transitions leaving from the initial state.

Figure 2.6a and Figure 2.6b show the initialization of the trellis corresponding to the model in Figure 2.5 - stochastic interpretation.

The iterative step in the forward supervisor strategy is the following.

1) For all the states in the i-th column of the trellis do

For all the non EMPTY transitions leaving from the

current state do

begin

compute the contribution of the current transition

update the scores of the terminal state of the

transition in the (i+1)-th column of the

trellis

end

2) For all the states in the (i+1)-th column of the trellis do

For all the EMPTY transitions leaving from the

current state do

begin

compute the contribution of the current transition

update the scores of the terminal state of the

transition in the (i+1)-th column of the

trellis

end

The updating phase substitutes the score associated with a state q_i in the trellis with value score(q_i) = combine(score(q_i),c(q_j,q_i)) where c(q_j,q_i) is the contribution of the transition from state q_j to q_i. The function "combine" is a function which combines the previous score associated with a state with the incoming one and propagates the contextual information from one state to another. This function must be supplied by the user. For example, in the stochastic interpretation of the strategy, the function is the maximum if we want to perform a maximum likelihood evaluation. If we want to calculate the sum of the probabilities of all the paths then the sum have to be used. In the interpretation of a distance the function to be used is the minimum.

Figure 2.7a and Figure 2.7b show the first iteration of the model in Figure 2.5 after the initialization and in Figure 2.8 the probabilities are reported - the combination function is the maximum and the corresponding best path is propagated as contextual information from state to state.

The same strategy can be applied in reverse and this is what is called supervisor backward strategy.

Initially at computational time (i.e. discrete step) 0 only the final state of the network belongs to the 0-th column. The associated score is supplied by the user

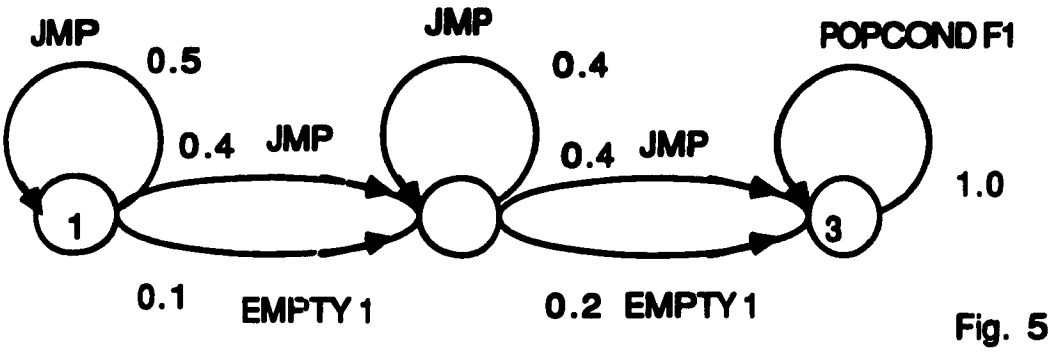


Figure 2.5: Example of EPN with stochastic interpretation

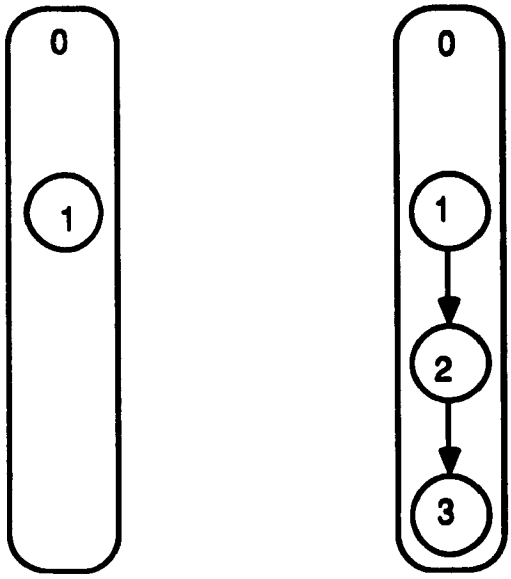


Figure 2.6a

Figure 2.6b

Figure 2.6: Forward initialization of the trellis of the model in Figure 2.5

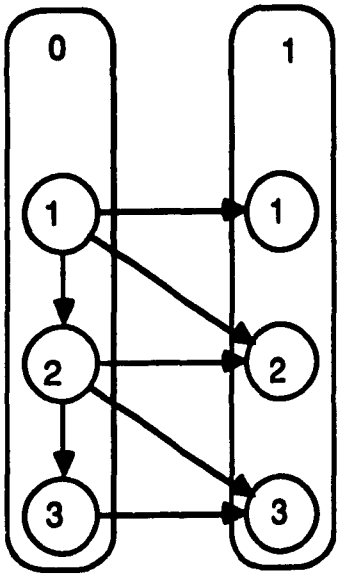


Figure 2.7a

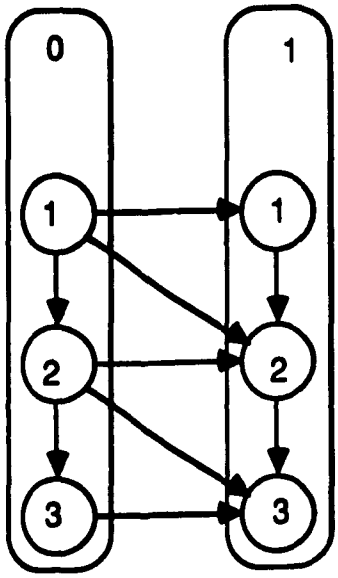


Figure 2.7b

Figure 2.7: Second forward iteration of the model in Figure 2.5

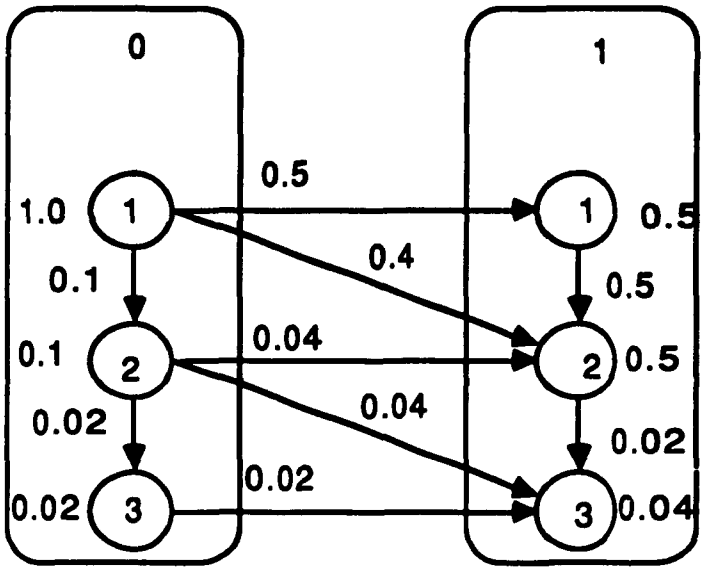


Figure 2.8: Forward measures of the trellis in Figure 2.7

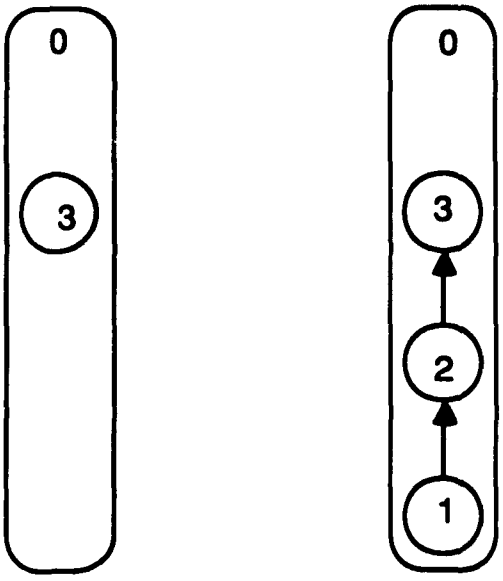


Figure 2.9a

Figure 2.9b

Figure 2.9: Backward initialization of the trellis of the model in Figure 2.5

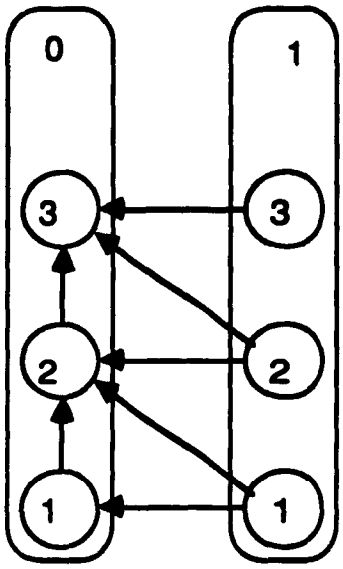


Figure 2.10a

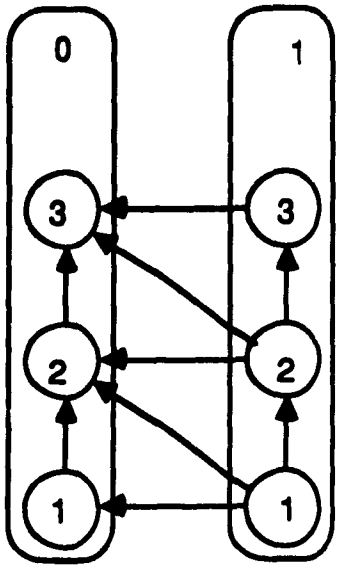


Figure 2.10b

Figure 2.10: Second backward iteration of the model in Figure 2.5

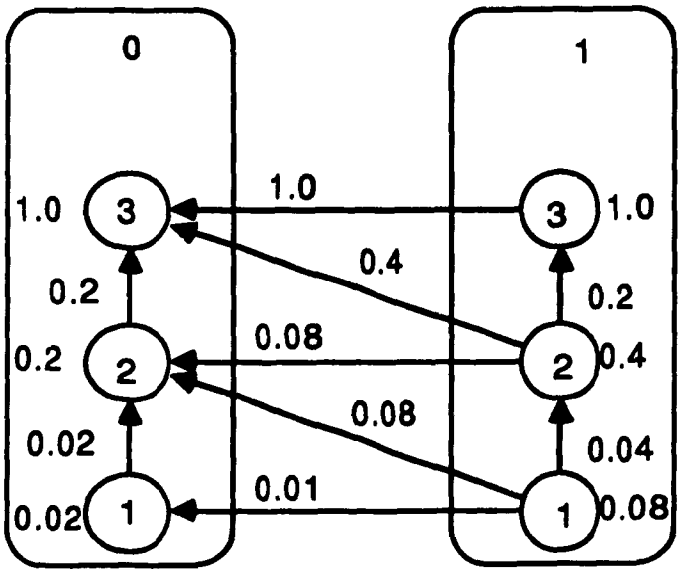


Figure 2.11: Backward measures of the trellis in Figure 2.7

as a parameter of the network. The score of the final state of the network depends on the interpretation of the reasoning process. If a stochastic approach has been taken then a reasonable value for the final state is 1, and so on.

The column 0 of the trellis would be subsequently completed by the states and contributions which arise from EMPTY transitions ending in the final state.

The trellis after the initialization is reported in Figure 2.9a and Figure 2.9b. The iterative step in the backward supervisor strategy is the following.

1) For all the states in the i-th column of the trellis do
    For all the non EMPTY transitions ending in the
        current state do
    begin
        compute the contribution of the current transition
        update the scores of the starting state of the
            transition in the (i-1)-th column of the
            trellis
    end

2) For all the states in the (i-1)-th column of the trellis do
    For all the EMPTY transitions ending in the
        current state do
    begin
        compute the contribution of the current transition

update the scores of the terminal state of the

transition in the (i-1)-th column of the

trellis

end

The updating function becomes then score(q_j) = combine(score(q_j),c(q_j,q_i)) where c(q_j,q_i) is the contribution of the transition from state q_j to q_i.

Figure 2.10a, Figure 2.10b show the first iteration and Figure 2.11 show the values when the combination function is the maximum and the best path is propagated as contextual information from state to state.

The complexity of the supervisor strategies is linear with respect to the trellis size which is considered as the number of states times the length of the trellis (computational steps).

2.3 - THE LANGUAGE

The EPN description can be considered as an A.I. programming language. Such a programming language corresponds to the actual input to the computer, while graphical description is more used for human analysis. The language definition has been implemented as a library of interface routines toward the Pascal compiler. The language is not directly compiled into some machine code. The user of the library (an EPN programmer) typically wants to parse and execute a set of EPN's. Such functions are accessible through any language which is

compatible with standard Pascal parameter passing mechanisms. The EPN can be described in the following format.

```
FUNCBEGIN
<compulsory function declaration>

GLOBALBEGIN
<global declaration>

BUFBEGIN
<state buffer declaration>
DECLEND

{
      DEFPNBEGIN <net name>
      <initial state name> <final state name>
      <epn declaration>
      {
            DEFTRBEGIN <transition name>
            <starting state> <terminal state>
            <transition parameters>
            DEFTREND
      }+
      DEFPNEND
}+
```

<global declaration> ::= <declaration>

<epn declaration> ::= <declaration>

<transition parameters> ::= <measure> <transition declaration>
CONDBEGIN <condition> ACTBEGIN <action> END

<transition declaration> ::= <declaration>

<state buffer declaration> ::= <variable declaration>

<declaration> ::= <constant declaration>
<type declaration>
<variable declaration>
<procedure declaration>

<condition> ::= DEFAULT r | <statement>

<action> ::= PUSH i | POPABS f | POPCOND f | JMP | INHIBIT a r |
EMPTY | EXPAND i | CREATE <transition> |
DELETE <transition> |
SET_PARAM <transition> <transition parameters> |
GET_PARAM <transition> <transition parameters> |
<statement>

<transition> ::= <transition_name> <starting state> <terminal state>

The scoping rules of the EPN are the following: an identifier referenced in a condition or action is sought in the transition declaration and in the state buffer declaration. The transition level includes transition declarations and state buffer declarations together at the same level. Therefore identifiers which appear in the state buffer declaration and in the transition declaration must be different. If an identifier is not found at the transition level, then subsequently the <epn declaration> and the <global declaration> are examined. If the identifier is still not found, then an undeclared identifier error is issued.

Procedures and functions defined in the function declaration cannot be referenced in any part of the network; they are used internally by the EPN supervisor.

A number of AI problems are reviewed in the following sections and their solution with EPN is shown.

2.4 - EPN AND AI STRATEGIES

The computational power of the EPN model is the same of linear bounded automata (refer to section 2.4.9). Furthermore the possibility of defining executable actions which are pieces of code gives the EPN the same power of conventional high level languages (HLL). Although all AI strategies can be implemented in the EPN formalism, some strategies are very easy to describe by simply tuning the parameters of the EPN. In the following sections some examples will be presented.

2.4.1 - PLANNING

A plan is some finite structure over a set of actions. A sequential plan is a finite sequence of actions. A plan defines a set of sequences of world states, called the behaviour of the plan. We can obtain one of these behaviours by executing the plan.

In general, the planning problem is the following: given an initial situation I_S and some desired condition (goal) G to be achieved we need to construct a plan that will guarantee that a state satisfying G can be achieved provided only that the plan is executed in a state that satisfies I_S. There are several known algorithms to

generate plans (planning), [11], [69], [89]. The EPN planner is a maximum likelihood strategy with no backtracking. In order to perform a planning algorithm all the probabilities are set to 1.0. Conditions are set to true if standard planning is desired. In the case of conditional planning, the conditions are activated and represent the conditions of the plan.

When the concurrency of several plans is required to achieve a goal the standard planning systems provide a structure of split and join to explicitly control cooperative or competitive plans. In the EPN such an explicit structure is not possible because of the interpretation of the network according to the data driven model strategy. Nevertheless, the concurrency or competition of goals is achieved using EPN built-in functions. Splitting a plan is the normal operation which is performed when there are more than one transition leaving from a state. Each transition starts an independent subplan. The difficult problem is to join different partial plans. The join is implicitly realized by the POPCOND action of a subnetwork, which then represents the operation of joining concurrent plans.

Wilkins's SIPE (System for Interactive Planning and Execution Monitoring [89]) performs the following operations:

- hierarchical
- forms partial plans
- allows limited parallelism
- uses constraints to increase search efficiency
- allows reasoning about resources

- provides reasonable deductive power
- allows conditional plans
- monitors execution
- can replan after execution failure

The EPN can perform all these activities except to replan after execution failure, because the EPN always finds a plan: it is the one with highest score. If there is no way to get to the desired goal (i.e. there is no plan satisfying the constraint) a system error is issued, but no replanning is performed if this was not previously embedded explicitly in the planning strategy.

Under certain conditions, the order in which the set of applicable rules is applied to the database is unimportant. A.I. systems of that kind are called commutative systems. The properties which characterize a commutative system are defined in [66].

In some A.I problems the most natural solution involves non commutative systems. The typical solution is a sequence of actions. Planning is this part of A.I. World state can only change through the occurrence of some event or action.

An event or action is a relation on states representing all possible occurrences of the event. This relation is usually assumed to be functional (i.e., events are viewed as transition functions on states). Events are represented by EPN states, the functional relation between events is the transition function on EPN states.

If we had a number of actions to be performed, we would have quite a number of conditions to specify that certain actions do not change the values of certain predicates [58]. The problem of keeping track of the consequences of performance of an action in a representation world is known as the frame problem, referring to the movie sequences of frames in which the background doesn't change whether the image is moving.

In the EPN the contextual information which is propagated from state to state is considered as the background, and it is the role of every action to change some part of such information. This constitutes a sort of procedural approach to the frame problem, with respect to a declarative approach (add list and delete list) like in STRIPS [66].

2.4.2 - EVIDENTIAL REASONING

Evidential reasoning has been used in A.I. to represent and manipulate incomplete and imperfect knowledge. Uncertainty arises from missing or erroneous data, missing or erroneous rules and incorrect modeling. Evidential reasoning removes some of the probabilistic axioms and gives rise to other theories and formulas. The best known approaches to manage such uncertainty are the certainty factor (CF) model [82] used in medical systems like Mycin - and later adapted for Emycin systems -, Dempster-Shafer theory [79] and Fuzzy Logic [91]. Certainty approach uses the concept of Measure of Belief (MB) and Measure of Disbelief (MD) to evaluate a certainty factor (CF). P(H) is the belief of an expert in hypothesis H and P(H/E) is the belief in H based on some

observation E. The definition of MB and MD can be found in [74].

The CF is represented in a EPN formalism by the value returned by function f_i. The measures of belief and disbelief are stored in the state buffer which is propagated. The EPN supervisor then finds the sequence of transitions (rules) that support a certain goal (final state) with a certain criterion matched on the certainty factor. For example, it seems to be reasonable to choose the highest certainty factor. Other criteria may be applied as well.

The Dempster-Shafer belief functions, like CF and Bayesian functions, assign numerical measures of belief in hypotheses based on an observed evidence. The D-S combination rule includes the Bayesian and CF functions as special cases. Another consequence of the generality of the D-S belief functions is avoidance of the Bayesian restriction that commitment of belief in a hypothesis implies commitment of the remaining belief in its negation. The D-S measures of belief assigned to each hypothesis in the original set need not sum to 1 but may sum to a number less than 1. Refer to [79] for details. In the EPN D-S measures can be propagated through the state buffer and the transitions represent the strategy of evidence gathering and reasoning.

2.4.3 - PROBLEM SOLVING

The EPN can perform problem solving by relating a state in the network as a state in the problem space. The network supervisor performs then a search in

such a space. Expansion of states at the same level (breadth first search) is achieved through the creation of transitions in the same subnetwork, while successive expansion of states at lower hierarchical level (depth first search) is achieved though PUSH and hierarchical nesting. Pruning is introduced through conditions. The EPN supervisor implicitly performs a breadth first search, unless otherwise instructed by PUSH actions. Creating transitions and states is achieved by using the structure modifying built-in functions (see section 2.5).

An EPN state can also represent a strategy state. In this case, the problem can be interpreted as a planning problem with finite state space. This case is detailed in section 2.4.1.

Among the many problem solving paradigms, hypothesize and test will be considered in the next sub-section.

2.4.4 - HYPOTHESIZE AND TEST

The hypothesize-and-test strategy is the simplest A.I. paradigm. The method consists of generating all possible solutions in the search space and testing each solution until one which satisfies a goal condition is found. The basic algorithm is the following:

1) Generate a possible solution. This means generating a particular point in the problem space.
2) Test to see if this is actually a solution by comparing the chosen point to the

goal conditions

3) if a solution has been found, quit. Otherwise return to step 1

The EPN corresponding to a hypothesize and test paradigm is reported in Figure 2.12.

Obviously the set of possible solutions must be finite and for practical reasons its cardinality should be kept small. If the number of hypotheses to be tested is reasonably small, the strategy can be explicitly declared in the EPN formalism. If we think of the network in terms of strategies, one state in the network corresponds to a particular strategy. Therefore the hypothesize and test paradigm has to be read as "hypothesize a strategy to solve the problem" and then "verify if the problem is solvable under that strategy". In this context the number of possible strategies that can be applied to solve a problem has no relation to the size of the search state space for a particular strategy. In general this number is fairly small and can be used in the EPN formalism with explicit declaration.

When the size of the problem becomes bigger and we want a dynamic creation of hypothesis and tests, actions that modify the structure and the parameters of the network have to be used. This dynamic generation of EPN transitions is encountered also in standard problem solving (see section 2.4.3).

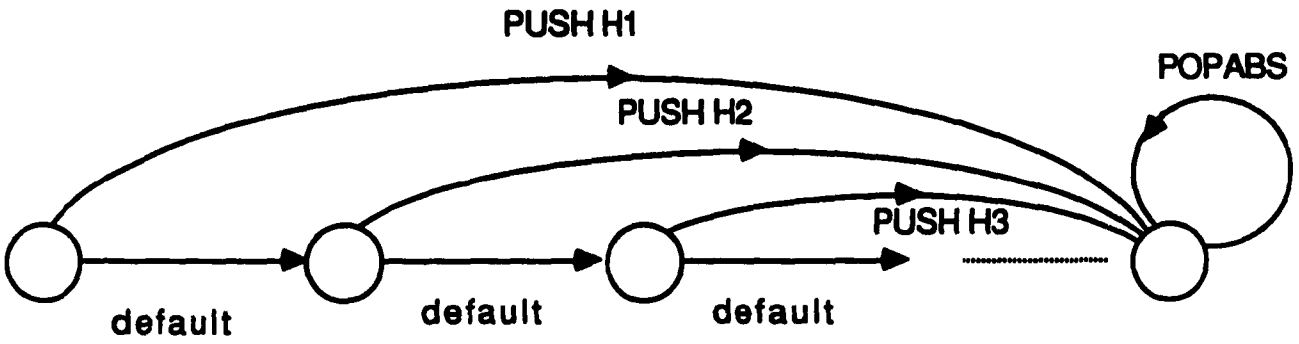


Figure 2.12: EPN for hypothesize-and-test paradigm

2.4.5 - NEURAL NETWORKS

Neural Networks [55], [78], can easily be implemented in the EPN formalism. The Multilayer Perceptron can be represented according to the EPN model. The interpretation of the parameters of a transition are the following: the weights w_i correspond to the measures P_i associated to the transitions. The values on the neural network states correspond to the scores associated to the EPN states. The function "combine" which combines an incoming contribution with the current score of a state corresponds to the sigmoid function.

Figure 2.13 shows an EPN which implements a neural network. Every transition at level u or v has the following parameters:

P_i is the weight w_i of the transition

$condition_i$ is defined to be always satisfied because the arcs do not have to be conditionally executed

$action_i$ is the built-in action JMP

Transitions at level t in Figure 2.13 set the input values, and transitions at level z propagate the output values to the final state.

Figure 2.14 shows an EPN for estimating the parameters of a neural net. The action LEARN modifies the measures P_i of transitions of type u,v according to the learning paradigm. The condition CONV tests for the reaching of the desired convergence.

PNeval:

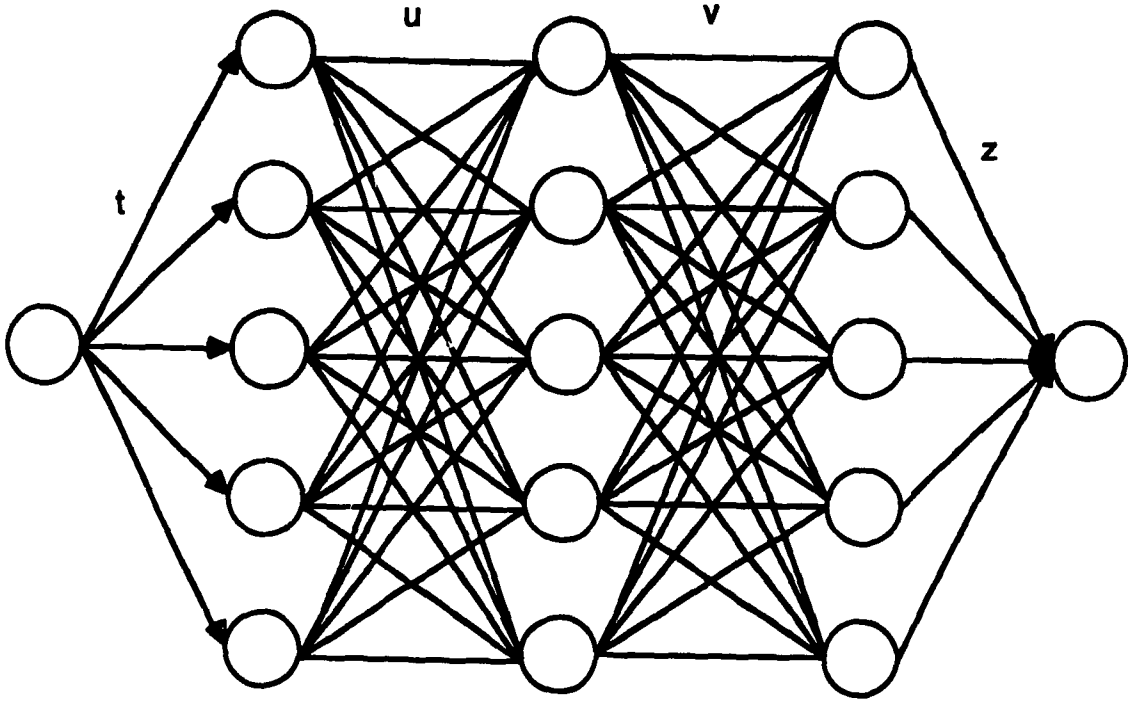


Figure 2.13: EPN that implements a neural network

PNlearn:

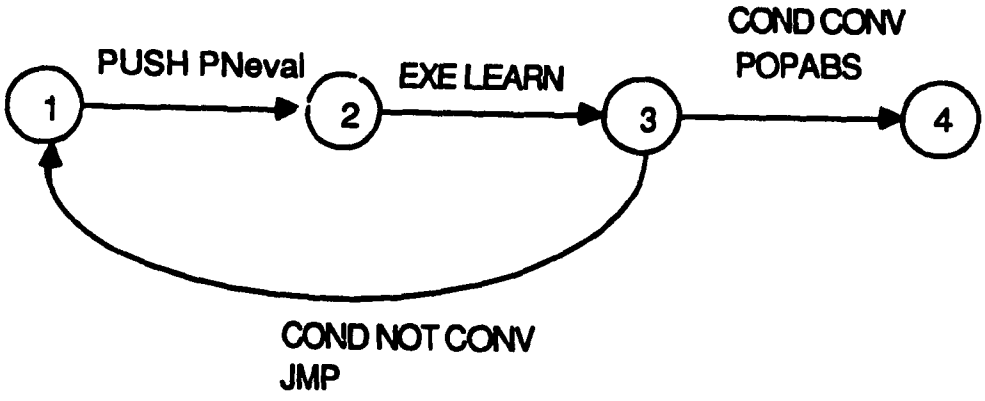


Figure 2.14: EPN that infers the parameters of a neural network

2.4.6 - DYNAMIC PROGRAMMING

Dynamic programming is a technique of algorithm design by which a problem is decomposed into a polynomial number of subproblems whose solutions can be considered as entries in a table. The table is filled with the solution of the subproblems without regard to whether or not a particular subproblem is actually needed in the overall solution. The structure of the table and the order in which the entries are filled are the characteristics of a particular dynamic programming algorithm. DP algorithms have been widely used in pattern recognition, in particular what is called DP matching [64]. Let us define a template pattern x as $x_1, x_2, \ldots, x_n$ and an input pattern y as $y_1, y_2, \ldots, y_n$. Let $a(x_i,y_j)$ be the cost of substituting x_i with x_j, $b(y_j)$ the cost of inserting y_j and $c(x_i)$ the cost of deleting x_i. The cost of matching x as $x_1, x_2, \ldots, x_i$ with y as $y_1, y_2, \ldots, y_j$ is computed on the base of the cost of matching $x_1, x_2, \ldots, x_{i-1}$ with $y_1, y_2, \ldots, y_{j-1}$, $x_1, x_2, \ldots, x_{i-1}$ with $y_1, y_2, \ldots, y_j$, and $x_1, x_2, \ldots, x_i$ with $y_1, y_2, \ldots, y_{j-1}$. If we define the cost of matching $x_1, x_2, \ldots, x_i$ with $y_1, y_2, \ldots, y_j$ as D(i,j) then in formulas:

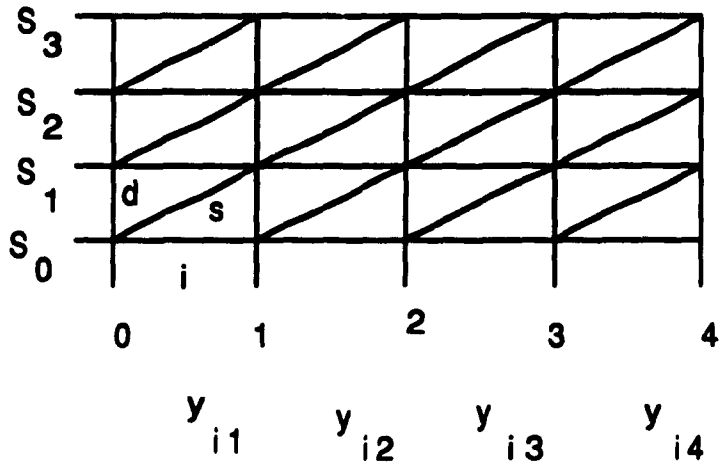
$$D(i,j) = f(D(i-1,j-1)+a(x_i,y_j), \quad D(i-1,j)+c(x_i), \quad D(i,j-1)+b(y_j))$$

If we assume that f = min then the formula becomes:

$$D(i,j) = min(D(i-1,j-1)+a(x_i,y_j), \quad D(i-1,j)+c(x_i), \quad D(i,j-1)+b(y_j))$$

If we identify i as the state number in the EPN and j as the trellis column of the EPN supervisor the cost D(n,m) can be computed by the EPN in Figure 2.16. The

Figure 2.15: Dynamic programming trellis



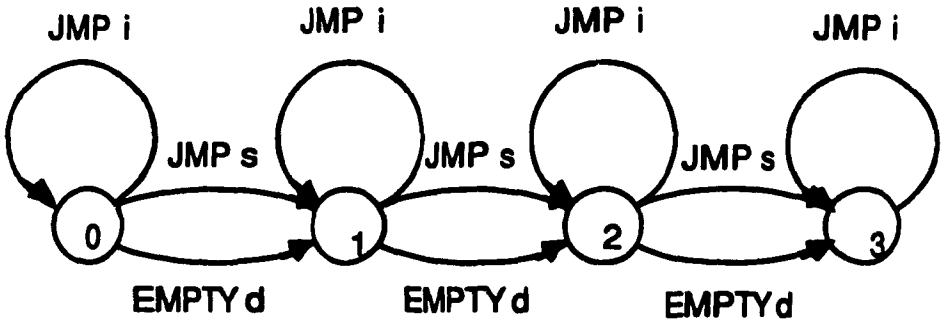


Figure 2.16: EPN that performs a dynamic programming pattern matching

corresponding Dynamic Programming trellis is reported in Figure 2.15.

2.4.7 - NONMONOTONIC REASONING

The importance of nonmonotonic or default reasoning has been discussed in many papers [29], [59], [73]. Nonmonotonic reasoning is important because it models the human way of reasoning in which we deal with incomplete information, with assumptions and with defaults that are considered true unless the opposite has been proven.

The EPN presents a nonmonotonic behaviour thanks to the inhibitory arc. A transition which is ruled by an inhibitory arc is normally executed except when the inhibitory condition is satisfied. In this way it is possible to represent default rules and hierarchical inheritance with exceptions.

With regard to the EPN the logic is somewhat different. In classical nonmonotonic logic, the database is considered to be time independent - i.e. a fact, which represents a certain belief is retained unless explicitly canceled (in some model). In the EPN the trellis structure corresponds to an historically dynamic set of beliefs, in which every fact that is not explicitly supported at every instant is removed from the database. This allows the user to decide whether or not some backtracking is needed or desired. In a classical Truth Maintenance System (TMS) as soon as a "justification" of a default condition is no longer satisfied, the TMS returns the system to a consistent state (i.e. with no

contradictions) according to different criteria (elimination of the facts which creates the inconsistency together with all the facts derived from them. In the EPN the context is different. Since the states normally disappear at every computational step, to retrieve and kill a path starting from a state which represents a fact which is no longer true may be meaningless. At the time when the state was active the fact was true and since the EPN represents a strategy it is perfectly correct that at that time and context that path had been chosen. In a classical TMS system all the facts are global. In the EPN the states carry some contextual and local meaning. In this context the nonmonotony is applied. If desired nothing prevents us from building global facts in the global "viewpoint" and reason about them and decide to run a dependencies directed backtracking algorithm on the basis of the current EPN active states. In this way the EPN would implement a problem solving plus TMS strategy and then backtracking would become an operator which is tied to a particular phase of the strategy (i.e. a particular transition).

The particular structure of the EPN causes some difficulties in defining the theory of the nonmonotonic EPN in the sense that:

1) the presence of an inhibitory fact does not affect the current reasoning unless both the inhibitory fact and the starting state of the inhibited transition are presently contained in the current column of the trellis. Therefore some classical problems of nonmonotonic reasoning are expressed differently in the language of the EPN.

2) So far the EPN does not perform any backtracking in the sense of a classical truth maintenance system [16], [27], but as shown in the example of Nautilus (Figure 2.17) taken from [30], a similar result can be achieved.

3) An important difference with classical default systems is that the facts in the database have attributes and a score. Attributes allow the system to perform inheritance with exceptions and scores select, in a non-arbitrary way, one deduction from a set of deductions in one extension. This solves the problem of how to choose or to switch from one extension to another during the reasoning process. Scores are also useful to deal with inconsistencies and contradictions which are inherent in a default logic system.

The EPN allows that a fact and its negation are simultaneously present in the database since possibly they carry different scores. An example of inheritance with exceptions is presented in Figure 2.17.

The EPN allows an explicit control strategy to take place. This means that what was implicit in other systems or left to a not well defined problem solver is now explicit. This reduces the need for backtracking since it is up to the system designer to model the strategy in such a way that either backtracking is explicitly applied or inconsistencies have to be accepted. Furthermore, nonmonotonic reasoning is applied in a well defined context in which the behaviour of the network is in principle clearly understood and planned. The structure of transitions and markings implies that there exists a specific ordering among the transitions depending on the time of activation. This avoids the problem that

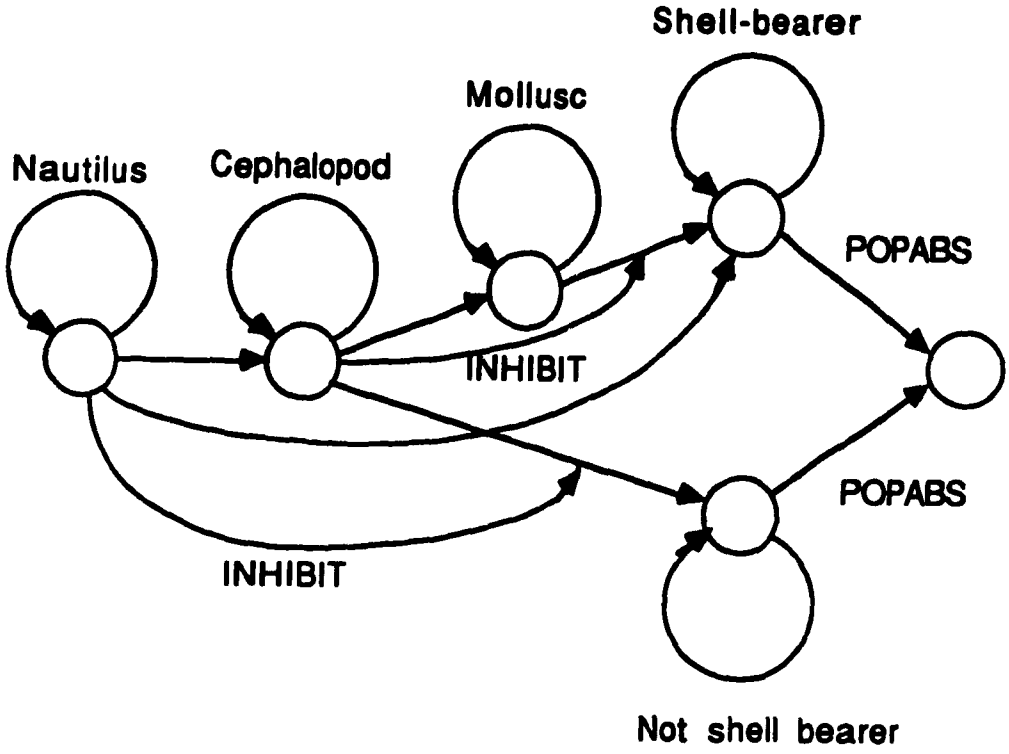


Figure 2.17: EPN for inheritance with exceptions

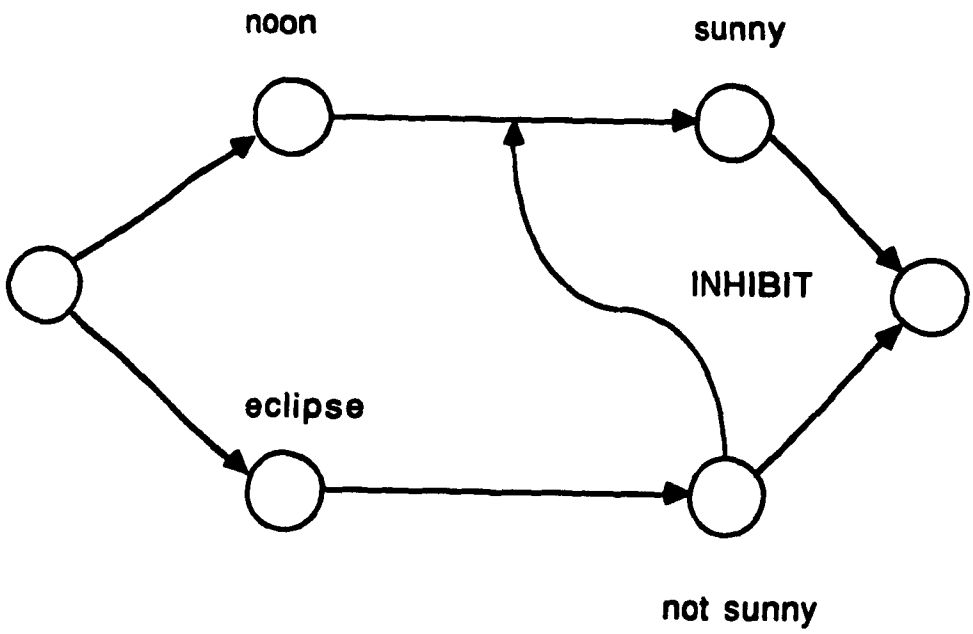


Figure 2.18: EPN with inhibitory actions

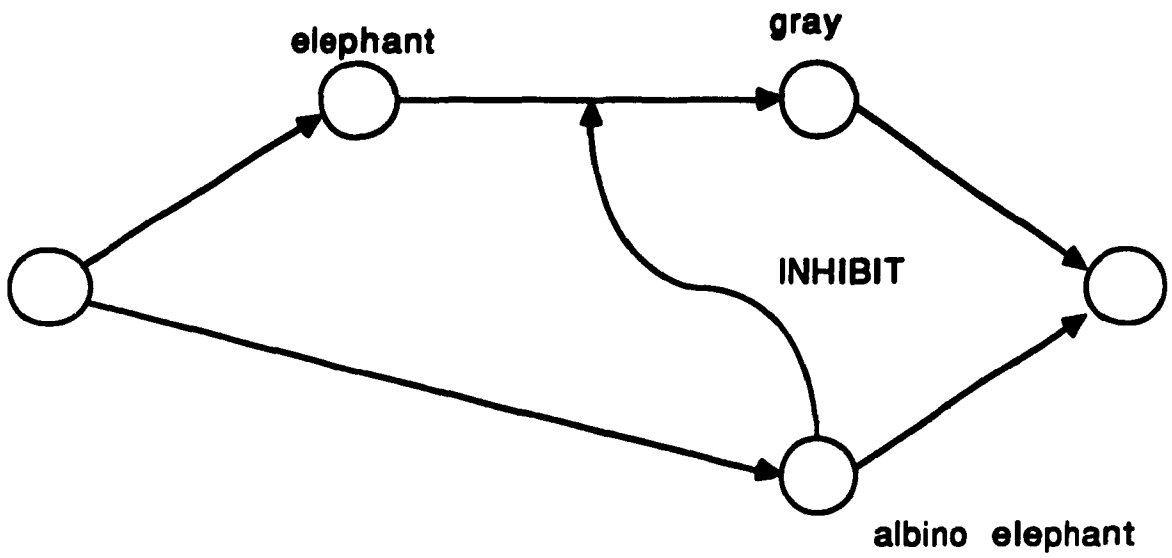


Figure 2.19: Example of EPN for inheritance with exceptions

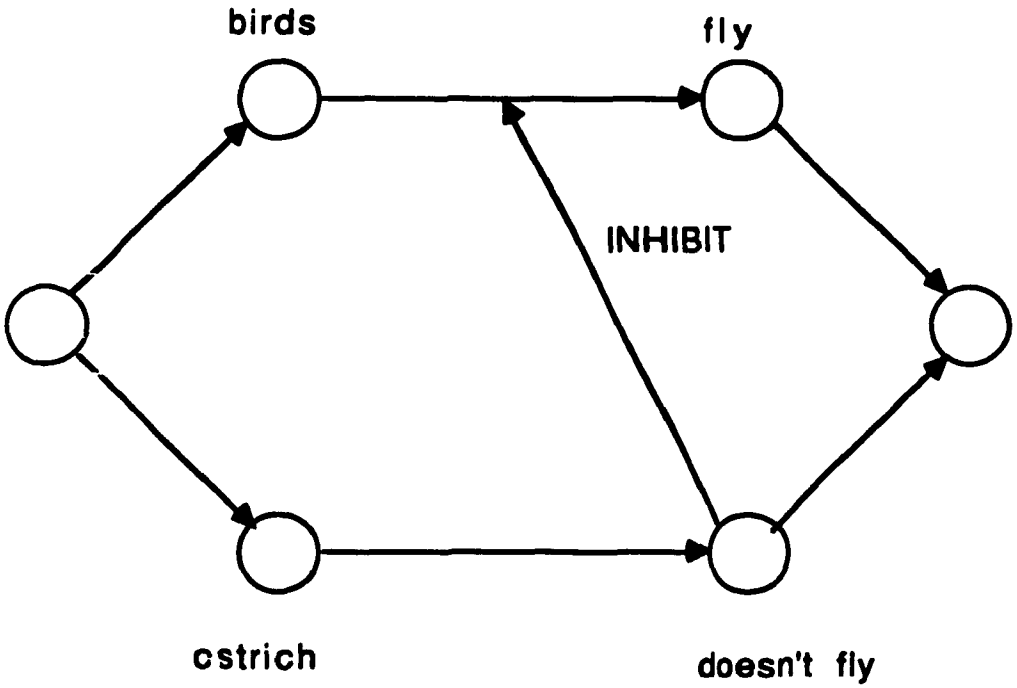


Figure 2.20: Example of EPN for inheritance with exceptions

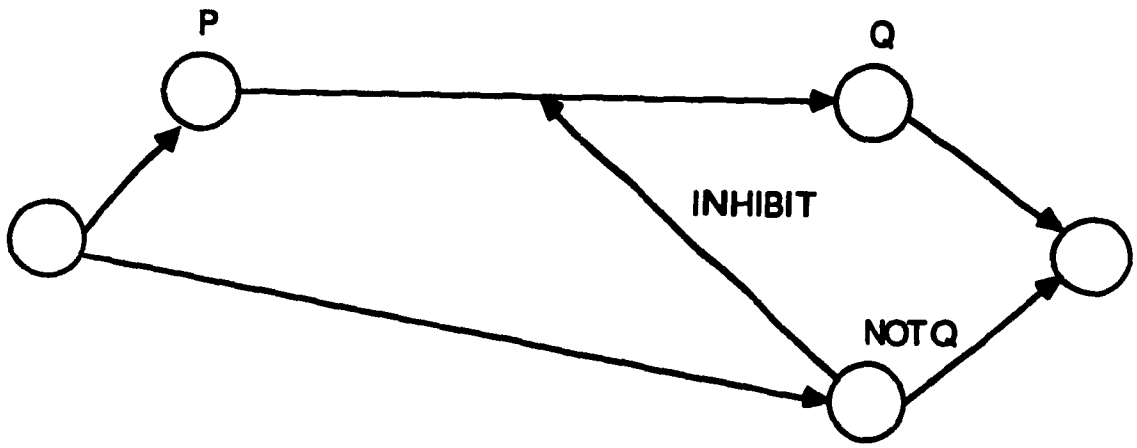


Figure 2.21: EPN for normal default logic

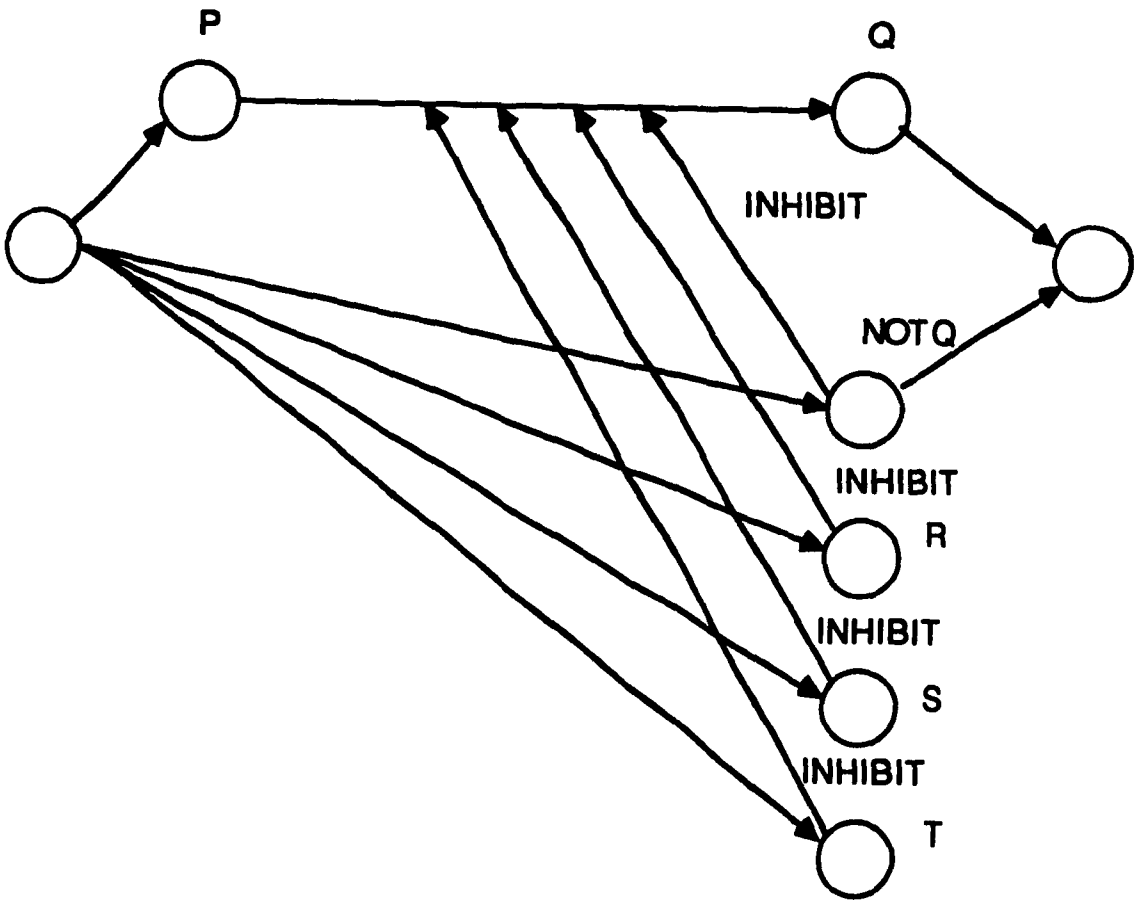


Figure 2.22: EPN for seminormal default logic

appears in certain networks of having different inheritance depending on the (unknown) order of parallel computation of transitions.

However some direct comparison can be made. Suppose that the following is a set of default assertions in the notation of [29]:

noon,sunny –> sunny

noon

eclipse –> ~sunny

eclipse

The corresponding network is represented in Figure 2.18. Three examples of inheritance with exceptions are shown in Figure 2.17, Figure 2.19, and Figure 2.20. They correspond to the following assertions:

a) Molluscs are normally Shell-bearers

Cephalopods must be Molluscs but normally are not Shell-bearers

Nautili must be Cephalopods and must be Shell-bearers

b) Elephants are normally gray

Albino elephants are not gray

Albino elephants are elephants

c) Birds normally fly except ostriches

Normal default networks [29] correspond to a structure like the one in Figure 2.21 and an example of a seminormal default network is represented by the network in Figure 2.22.

2.4.8 - RULE BASED SYSTEMS

Rule based systems consist of:

a) a set of rules in which the left side describe the applicability of the rule and the right hand side indicates the consequence of applying a rule

b) some databases containing facts (elements of the database) and information about the reasoning process

c) a control strategy which specifies the order of application of rules and the criteria of solving the conflicts which arise when more than one rule are applicable at the same time.

Rule based systems are easily implemented in the EPN formalism. Rules are represented by transitions. Facts correspond to states in the network. If the state appears in the current column of the trellis it means that the fact is asserted and it is the precondition to the firing of a rule (transition). The initial state corresponds to the initial situation of the database, while the final state corresponds to the goal which has to be proven. Rules are practically clustered in the sense that only the transitions which are defined as going out of a state can be possibly taken. This is equivalent to clustering rules to define context of applicability. Furthermore the network provides inherently numerical attachment to rules (scores), procedural-attachments, local and global scopes.

Classical control strategies in rule based systems are Forward and Backward chaining. These two strategies are embedded in the EPN Supervisor.

2.4.9 - SYNTACTIC APPROACHES

The EPN has the power to represent strategies which can be described by Context Sensitive Languages. In particular the EPN can implement a linear bounded automaton defined as in [39] and therefore accept Context Sensitive Languages. The EPN states and actions represent the finite control and the tape can be thought as an EPN global variable.

Recognizers for languages at a lower level in the hierarchy defined by Chomsky can be defined by the EPN model. Pushdown automata and finite state automata are easily described in the EPN formalism. Pushdown automata can be simulated by identifying the input tape and the pushdown stack as EPN global variables and the finite control with EPN states and actions. Finite state automata can be simulated by considering the input tape as an EPN global variable and the finite control as EPN states and actions. Details and applications of syntactic pattern recognition can be found in [32], [34].

2.5 - LEARNING

An important ability of intelligent agents is to adapt to new situations, rather than simply doing as they were told to do. In the EPN two different kinds of learning can be defined: learning by parameter adjustment - a kind of specialization - and structural learning. Learning by parameter adjustment is an iterative method that estimates the measures associated with a transition on the

basis of the past experience. The new parameters are used to formulate and solve new problems.

An example of learning by parameter adjustment is presented with the assumption that the function "combine" (see section 2.2) is the maximum and the best path is propagated as contextual information. If the network or subnetwork is monotonic (i.e. the inhibitory arc is not used) then the learning algorithm is optimal in the sense that the estimates converge to those values of parameters which maximize the scores of all the correct solutions of the problems presented as learning samples. The structure of the network (i.e. states and transitions) is supposed to be known. The parameters are estimated as function of this structure.

Let $S = S_1, S_2, \cdots, S_m$ be the best paths corresponding to a certain training set, and $S_i = t_{i1}, t_{i2}, \cdots, t_{in}$ be the sequence of transitions belonging to the i-th best path. Let us define

$$N(t) = \sum_{i=1}^{m} \sum_{j=1}^{n} \delta(t_{ij},t)$$

as the number of times that transition t has been used to process the training set, where

$$\delta(x,y) = \begin{cases} 1 \text{ if } x=y \\ 0 \text{ if } x\neq y \end{cases}$$

The expected frequency f_t of transition t can be estimated as

$$f_t = \frac{N(t)}{\sum_{t'/L(t)=L(t')} N(t')}$$

Another point of view of learning by parameter adjustment is that of changing the measures of the arcs to influence future strategies according to the current one.

Structural learning is directly used in A.I. approaches like problem solving, hypothesize and test paradigm, and whenever the topological structure of states and transitions is intrinsically dynamic. The structure of the network can be inferred by using a set of examples. The first step is to define a language Σ whose alphabet is the alphabet of condition-action pairs. The training set is represented as a sequence of vectors whose components are condition-action pairs a_{ij}. Let $L = L_1, L_2, \cdots, L_m$, where

$$L_1 = \{a_{11}, a_{12}, \cdots, a_{1n_1}\}$$

$$L_2 = \{a_{21}, a_{22}, \cdots, a_{2n_2}\}$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$L_m = \{a_{m1}, a_{m2}, \cdots, a_{mn_m}\}$$

be the training set. An algorithm for the inference of finite state automata [9],[34] can be used to infer the EPN structure which correspond to that training set. Structural learning, rather than language driven, can also be rule base driven, plan

driven and so on.

The generalization power of the learning approach is that of the underlying algorithm. In the proposed examples, the function f_t converges through iterations to a value that unfortunately is optimal only in a local way. The estimated value depends therefore on the initial value of p_t. The language driven structural learning contains a parameter k which controls the generalization power of the inferred automaton. The learning algorithm can be tuned to recognize finite languages which consist only in the training set up to more general languages, and finally to Σ^*.

2.6 - CONCLUSIONS

A new language for the description of integrated numerical and symbolic computations has been introduced. An overview of straightforward applications to implement AI paradigms has been provided. An application to ASR will be presented in detail in the next chapters.

CHAPTER - 3

3 - A MODEL FOR COMPUTER PERCEPTION OF SPEECH

It will be shown in chapter 5 how EPN can implement a perceptual model.

The speech signal $x_1(t)$ is **generated** by a discrete and finite sequence of actions

$$A = a_1(t_1)a_2(t_2)a_3(t_3).....a_k(t_k).....a_K(t_K) \tag{3.1}$$

where $a_k(t_k)$ denotes an action ending at time t_k; $a_1(t_1)$ represents the silence preceding the beginning of a sentence.

When a person reads a sentence S, a relation

$$R_1(S,A) \tag{3.2}$$

is applied which produces A. The relation R_1 may depend on the speaker, his/her mood, state of health and history. As R_1 may produce several As for the same S, probability distributions for all the possible As can be derived using a generative model.

The speech signal $x_1(t)$ is generated by the sequence of actions A using another relation

$$R_2(A, x_1(t)) \tag{3.3}$$

R_2 depends on the anatomy of the speaker. Again, the same actions may produce different signals, because the speech production system is soft and its behavior is affected to some extent by the environment.

If the speaker does not read but generates a sentence from a set C of concepts, then a third relation is applied:

$$R_3(C,S) \tag{3.4}$$

R_3 may depend on the speaker and his/her culture. Statistical models can also be used for characterizing this relation.

The generation of $x_1(t)$ can be seen as the application of the following composite relation:

$$G=(R_3 o R_1) o R_2 \tag{3.5}$$

according to the scheme shown in Figure 3.1.

Recognition consists in applying the relations in the opposite direction. Unfortunately we have only a limited knowledge of these relations. We have used it for building speech synthesizers. We do not even know the alphabet $\sum_A \{a_k\}$ for the elements of A, although we know alphabets $\sum_C$ and $\sum_S$ for the elements of C and S respectively. Furthermore, signal $x_1(t)$ is affected by noise and is transformed into another signal $x(t)$ through the acoustic channel.

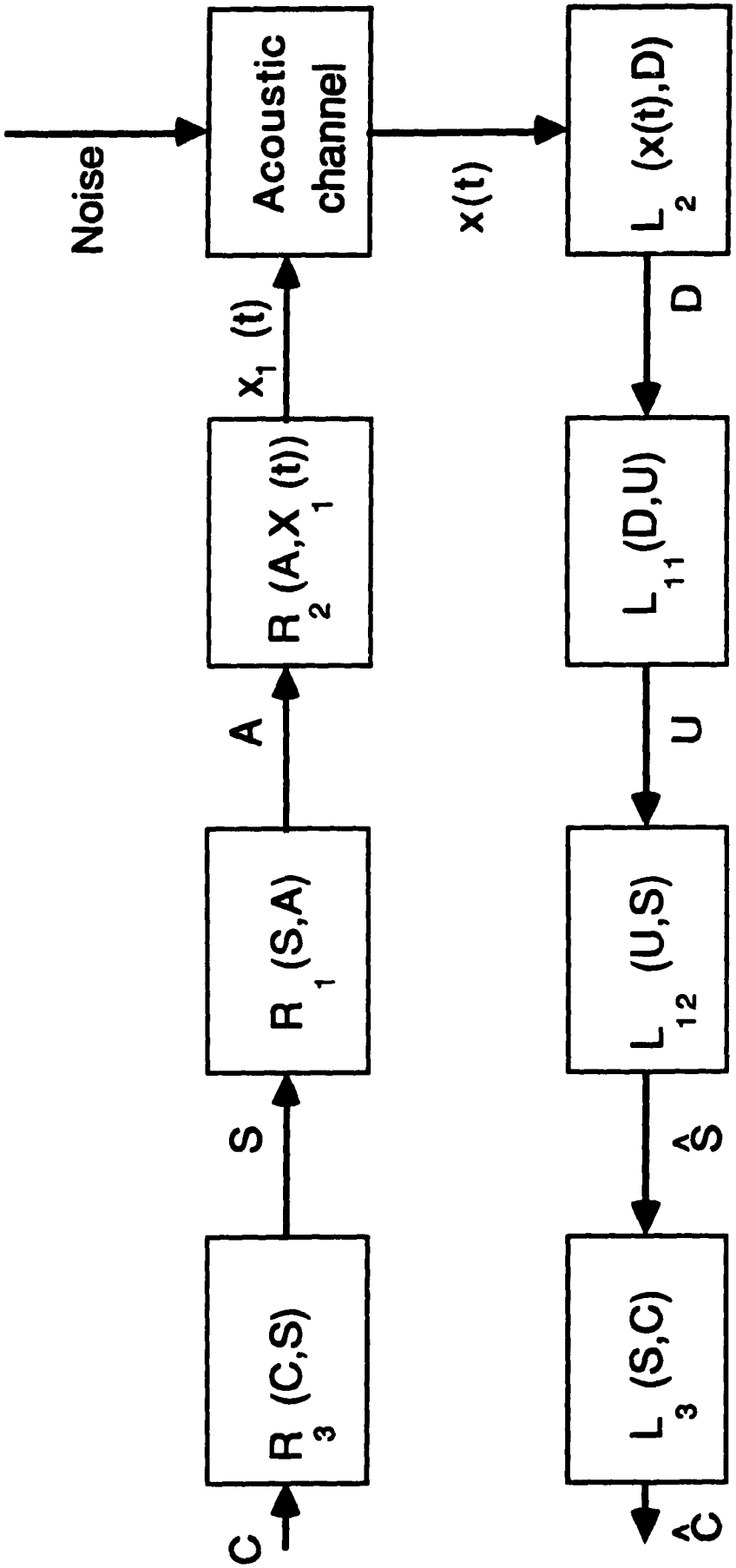
As we do not know $\sum_A$, nor we know R_2, we can characterize actions by **descriptions** of what they produce. According to this approach, the **perception** of $x(t)$ consists in extracting a sequence of descriptions:

$$D=d_1(\tau_1)\ d_2(\tau_2)....d_i(\tau_i)....d_I(\tau_I) \tag{3.6}$$

where $d_1(\tau_1)$ describes the silence preceding the beginning of the speech signal and $d_i(\tau_i)$ describes the segment of $x(t)$ between the time instants τ_{i-1} and τ_i.

Segments of D can be 10 ms frames or intervals of variable duration obtained by a segmentation algorithm like the one proposed in [20].

Figure 3.1: Speech Communication Channel Model



The descriptions D can be obtained by **perceptual actions** by analogy with the generative scheme. Perceptual actions, as well as generative actions, have to be defined and used according to a criterion of economy. That is, there must be a limited number of actions (operators) based on which a variety of networks of actions can be built.

Recognition can be seen as a combination of a relation

$$L_1(D,S) \tag{3.7}$$

that is the perceptual counterpart of relation R_1 used for speech generation, and a relation:

$$L_2(x(t),D) \tag{3.8}$$

that is the perceptual counterpart of $R_2(A,x(t))$

The relation $L_2(x(t),D)$ is **deterministic** in the sense that it can produce only one description D for a signal $x(t)$. Description D is a sequence of descriptive phrases. Each phrase can be of **fixed duration**, i.e., generated at constant time intervals, or of **variable duration**, i.e., generated for intervals of different length. If we want to maintain the analogy with the production model just outlined, D should be of variable duration because the articulatory actions (gestures) are of variable duration.

Descriptions must refer to parameters, morphologies and properties that are characteristic for a sound and exhibit low variances when many speakers, different microphones and environments are considered.

In practice, fixed duration models have been developed and tested with a considerable degree of success mostly in speaker-dependent systems. In one of the most successful systems developed so far [42], D is a sequence of symbols obtained every 10 ms by **vector-quantization** with a process that is speaker-dependent and context-independent.

Relation L_1 (D,S) has to capture two different types of knowledge. The first type of knowledge is a relation:

$$L_{11}(D,U) \tag{3.9}$$

between a sequence U of **Speech Units** (SU) and corresponding description D. There are speech units like the plosive sound /*b*/ for which a large variety of different descriptions D are perceived as the **same sound**. Relation L_{11} is many-to-one and it could be interesting to collect statistics of the elements of the universe of acoustic descriptions that produce the perception of the same linguistic sound. These statistics may represent distributions of acoustic patterns produced by a single or many speakers having the intention of producing the same sound. Statistics may also take into account characteristics of background noise.

The choice of SUs, for our purpose, has to be based on practical considerations as well as on theoretical ones. For example, for some purposes, units can be just phone classes or syllable classes.

The introduction of SUs is important because once a vocabulary $\sum_U$ of speech units has been chosen and effective relations between each $SU \in \sum_U$ and descriptions of acoustic properties have been established, large varieties of word

and sentence models can be built by **compiling** networks of SU models.

A second type of knowledge is a relation:

$$L_{12}(U,S) \tag{3.10}$$

where S is a linguistic entity like a sentence and U is a sequence of Speech Units. L_{12} can also contain statistics.

L_{12} may represent how different speakers may have different pronunciations of the same word. A stochastic model representing a word W in terms of SUs can be built.

An interesting possibility, which we explore in this thesis, is that of designing L_2 and L_1 **procedurally**, through actions to be performed on $x(t)$ in order to obtain D,U and S. Chapter 4 describes operators which compute relation L_2 and Chapter 5 contain details about different strategies corresponding to relation L_1 in different contexts.

Knowledge-based extraction and interpretation of signal properties has proven to be very effective when interpretation can benefit from contextual relations [65].

Descriptions D of different level of detail (depth) can be obtained depending on model expectations or the already available context. Feedback is also possible between relations, although it has not been implemented in the application described in this paper.

It seems that **variable depth descriptions** can be very useful in complex tasks where a preliminary selection of hypotheses has to be done based on robust but

simple descriptions and then a more detailed analysis has to be performed involving levels of depth depending on the competing hypotheses or on acoustic evidence.

The entire perception model can be represented by **extended procedural networks** which invoke subnetworks at several levels.

Building extended procedural networks is an activity of **conditional planning**. Elementary planning techniques and conditional planning are discussed in [33], [69], [86]. An introduction to the use of planning techniques for ASR is described in [21].

The most general extended procedural network has to operate along **two** dimensions using acoustic properties extracted in different time intervals and at different levels of detail. It will be shown in the following sections how these capabilities can be performed in the EPN model.

In order to perform variable depth analysis, a context in which the analysis is performed has to be defined. Algorithms were proposed in the past for segmenting continuous speech into Pseudo-Syllabic-Segments [20]. Although these algorithms have shown good performances in different tasks and for varieties of speakers, they were not error free in segmenting the speech signal into syllables. The principal reason for these errors was that segmentation was based only on **acoustic evidence. Acoustic Segments** (AS), obtained by segmentation algorithms based on acoustic properties, have to be treated as data rather than interpretations. Being based on acoustic evidence, ASs can be used for driving

and delimiting the extraction of more detailed acoustic properties or as anchors for lexical access in continuous speech.

Once an AS has been delimited, an Extended Procedural Network (EPN) is invoked to further segment it into **intervals** and to generate scored SU hypotheses on each interval. Scored SU hypotheses are used by word models that consider possible distortions, insertions and deletions.

Figure 3.2 shows a sort **performance model** for the word W.

For the sake of simplicity, the alphabet $\sum_U$ is made equal to the ARPABET (see. [47]).

In practice less detailed models suffice for recognizing the vocabulary defined in Table 3.1.

Chapter 4 and chapter 5 describe how the just introduced model can be implemented with EPN.

Figure 3.2: Performance model of the word W using the ARPABET for $\sum_U$

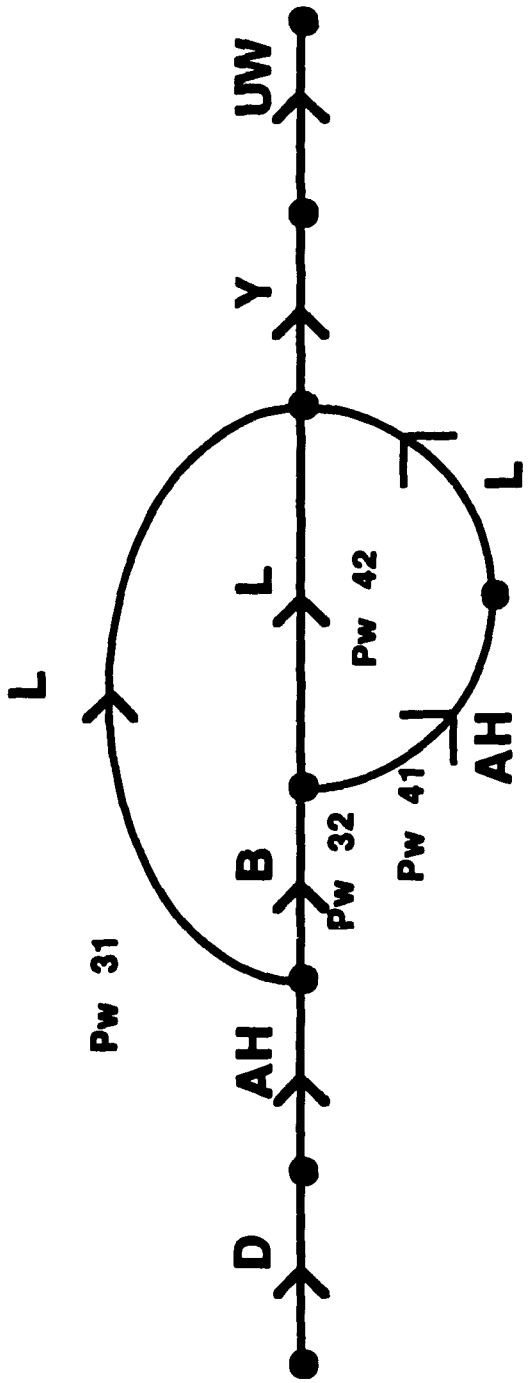


Table 3.1: Letters and digits vocabulary definition

Zero	One	Two	Three
Four	Five	Six	Seven
Eight	Nine	A	B
C	D	E	F
G	H	I	J
K	L	M	N
O	P	Q	R
S	T	U	V
W	X	Y	Z

CHAPTER - 4

4 - APPLICATION TO ASR: THE NETWORK OPERATORS

Before describing the recognition strategy, new ideas for speech analysis, that are suitable to be embedded in the flexible EPN paradigm, are introduced.

Section 4.1 deals with spectral lines, their extraction from the speech signal and their treatment. Section 4.2 introduces Markov models in the frequency domain and their relation to spectral lines. Section 4.3 presents the training algorithm, section 4.4 presents a diphthong recognizer based on the Markov models of section 4.2, and section 4.5 introduces a word recognizer based on neural nets memberships.

4.1 - ANALYSIS OF SPECTRAL LINES

4.1.1 - EXTRACTION AND DESCRIPTION OF SPECTRAL LINES

For spectrogram segments exhibiting narrow-band resonances, spectral lines are extracted from a time-frequency-energy representation of a speech unit using skeletonization techniques already used for image analysis [63]. These techniques have been adapted to spectrogram lines.

Skeletonization can detect a variable number of lines with different durations inside an acoustic segment, thus avoiding the errors and the difficulties of tracking formants [48].

Each spectral line is described by a vector of triplets (time, frequency, energy) that represents the lowest level (level-0) of a **time-frequency morphology taxonomy.**

It is worth mentioning that spectral lines extracted with skeletonization always contain formants when they are detectable with peak-picking techniques, but very often contain other lines. The system of lines obtained in this way is richer than the system of formants that can be tracked interactively on spectrograms and used for reconstructing understandable speech.

A recent paper by Kopec [48] attempts to track formants using Markov models. In the approach proposed in this thesis a set of lines is tracked that is redundant with respect to a set of formants. Distortions, insertions and deletions of spectral lines are taken into account in each SU model.

The motivation for such an approach is that spectral lines are significant acoustic properties but we do not know exactly which of them, if any, are not essential. We know that different speakers produce similar lines when they pronounce, for example, the same vowel. Relative frequencies and amplitudes between lines may vary from speaker to speaker in a limited range and bigger variations can be characterized as insertions or deletions. Distortions of relative line frequencies and amplitudes as well as insertions and deletions reflect inter- and intra-speaker variabilities and are described by statistical methods because their statistics are the only knowledge we can systematically acquire and generalize.

The above discussion is incomplete because spectral lines as extracted in our system cannot completely describe every type of speech units. In this thesis, we will limit our attention to spectral lines in vowels and diphthongs.

The skeletonization algorithm, described next, extracts spectral lines from the time-frequency-energy patterns obtained by considering the 0-4 kHz portions of spectra computed with the Fast Fourier Transform (FFT) algorithm applied to the preemphasized speech signal. A description of spectral lines is then obtained.

The Skeletonization Algorithm

The time-frequency-energy pattern for a given speech segment (see [20] for segmentation algorithm) generated by the FFT algorithm goes through 2 stages, namely, **thinning** and **preprocessing** before description. The pattern is thinned using the **Safe-Point Thinning Algorithm** (SPTA) described in [63].

There are two important restrictions imposed on the choice of the skeletonization algorithm for our application, namely:

1. connectivity of lines should be maintained by keeping the points at junctions,
2. excess erosion shouldn't be allowed.

The SPTA was chosen because it meets the above conditions.

Figure 4.1 shows an example of such a pattern for the diphthong /aei/ of /k/ before it is thinned and Figure 4.2 shows the thinned pattern.

In Figure 4.1 and Figure 4.2, time increases along the horizontal axis and each printed line corresponds to a 10 ms interval. Frequency is shown along the vertical axis. Intervals correspond to spectral peaks cut 6dB below the maximum. Energy of spectral peaks is coded by letters and digits. Letter B represents twice the energy represented by letter A; digit 0 represents an energy that is twice than the one represented by Z etc.

Preprocessing on skeletonized patterns is performed to discard all isolated, weak, and scattered points in the pattern. Preprocessing is carried out by applying an algorithm based on the strategy of **tracing continuity.**

The **Line Tracing Algorithm** (LTA) retains properties of collinearity, curvelinearity, continuity etc. present in the pattern. The significant lines in speech patterns are usually surrounded by lines which are less significant.

Thinning and preprocessing surface all significant and non-significant lines in the pattern and discard all scattered points.

LTA accepts the skeletonized pattern and applies an algorithm for smoothing.

The skeletonized pattern is a binary image which contains only **dark** and **white** points. The five-neighbours of a point P_i are defined to be the 5 points adjacent to P_i. A continuous line, l, exists between points P_0 and P_I iff there exists a path $P_0 P_1 \ldots P_{i-1} P_i \ldots P_I$ such that P_{i-1} is a neighbour of P_i for $1 \leq i \leq I$. A path between points P_i and P_{i+1} exists iff there exists at least one dark point among its neighbours. If more than one dark point exists among its neighbouring

Figure 4.1: Relevant spectral peaks in a pattern of the diphthong /aei/.

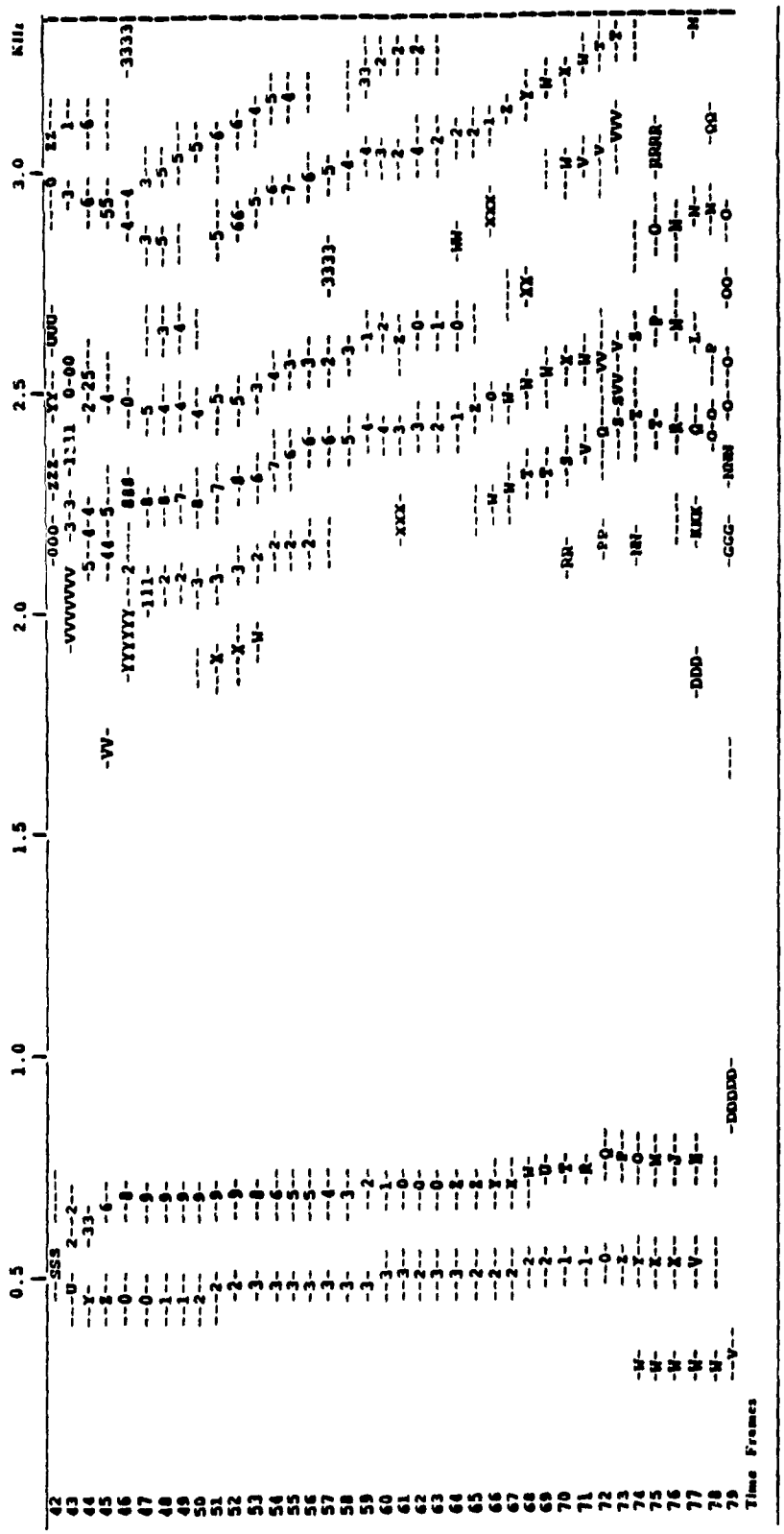


Figure 4.2: The pattern of Figure 4.1 after trimming.

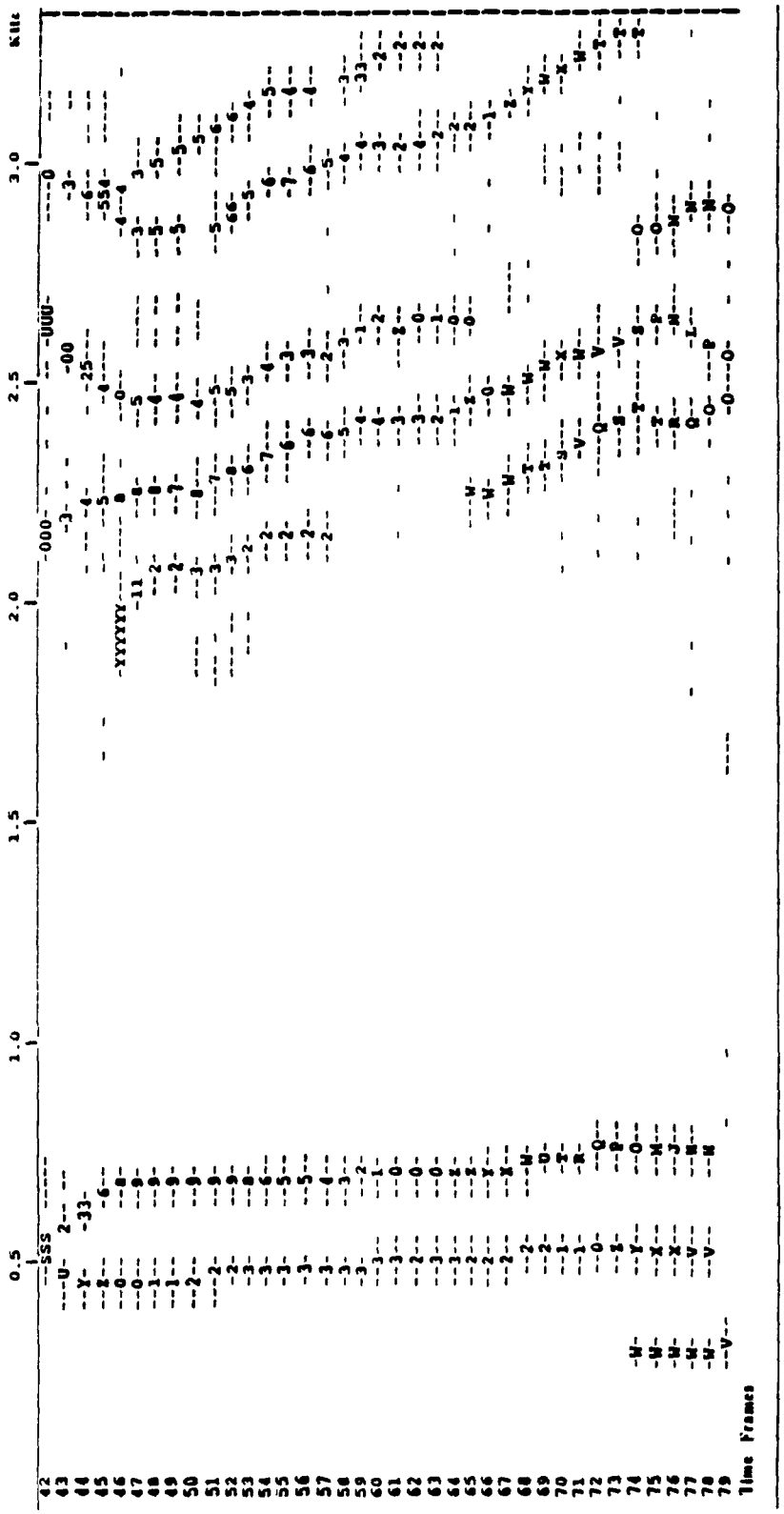


Figure 4.3: The pattern of Figure 4.2 after smoothing.

Table 4.1: The Line Tracing Algorithm.

```
line_tracing_algorithm (pattern:spectrogram; var vector:lines)
/ pattern is a binary image of the speech pattern    /
/ vector will have all detected lines in the pattern  /

begin
    set line counter, k = 0 ;
    for each row in pattern do
        begin
        for each column in pattern do
            begin
            set line_end = false ;
            while not line_end do
                begin
                look for dark_point, p, in pattern ;
                compute_neighbours [1], n, of point p ;
                if n =0 then                          /end_of_line found/
                    if rule1[2] then
                        begin
                        increment line_counter, k ;
                        accept current_line as k ;
                        set line_end = true
                        end
                if n = 1 then                  /1 neighbour for p/
                    begin
                    accept point p for line, k;
                    set point p in pattern as white;
                    set  new neighbour as point p;
                    continue tracing
                    end
                if n > 1 then                  /junction found/
                    begin
                    accept point p for line, k ;
                    set point p in pattern as white ;
                    set point p as strongest new neighbour ;
                    continue tracing
                    end
                end_while
            end_do
        end_do
end
```

1: neighbours, n, is computed as,

$$n = (i\text{-}1,j) + (i\text{-}1,j+1) + (i,j+1) + (i+1,j+1) + (i+1,j)$$

where i and j points to the location of p in pattern.

2: rule1 = true , if

$k(p) > \psi_1$ and $k(h) > \psi_2$

where, k is the k^{th} line

p is the number of points in line k

h is the height of line, k

ψ_1 and ψ_2 were empirically determined constants.

points n_{i0}—n_{i4}, then, the point n_j with the maximum energy is considered. If the maximum energy point is not unique, then the algorithm to find line l, is recursively applied to find the line which is the longest from point P_l. The algorithm written in Pascal-like notation is given in Table 4.1. Further details can be found in [68]. Figure 4.3 shows the pattern produced by smoothing the pattern of Figure 4.1.

The number of lines that appear in a pattern depends on thresholds that can be varied in order to have a desired effect. Our objective is that of keeping small the probability of loosing formant lines. On the contrary the methods for handling spectral lines that will be proposed in the following are well suited for taking into account redundant lines.

Various solutions have been investigated for reducing the number of redundant lines due to pitch effects. They include the possibility of using pitch synchronous FFT or cepstral analysis in selected time intervals and to use their results on a filter for lines generated with asynchronous FFT. Such filters are applied in such a way that a sufficient number of lines is kept in at least three frequency bands in which formants may be present.

4.1.2 - DESCRIPTION AND SEGMENTATION

Spectral lines can be described at several levels. At the lowest level, each line is described as an independent object whose relations with other objects (lines)

OVERVIEW OF TIME-FREQUENCY TAXONOMY	
level-0	each spectral line is described by a vector of triplets (time,frequency,energy)
level-1	each spectral line is described by a summary expressing its behavior and frequency band
level-2	quasi-stationary short-time segments of spectral line patterns are related to places of articulation
level-3	properties of the time evolution of spectral lines are described

Figure 4.4

are not considered. Higher level descriptions involve relations between objects (lines) both in the time and frequency domain. Such relations can be eventually structured in a time-frequency taxonomy. An overview of the taxonomy is shown in Figure 4.4.

At **level-0** of the taxonomy a spectral line is described by a vector V_j of triplets (t_{ji},f_{ji},e_{ji}) $(j=1,...J;i=1,...,I_j)$ where t_{ji} is a time reference in centiseconds, f_{ji} is a frequency value in Hz and e_{ji} is an energy value in dB. J is the total number of spectral lines in a pattern; I_j is the number of time frames (a time frame usually has a 10 ms duration) corresponding to the duration of the j-th line. The i-th sample of the j-th line is represented by (t_{ji},f_{ji},e_{ji}). The line bandwidth is not considered because it is in principle redundant and in practice difficult to estimate.

The information contained in vectors V_j can be further compressed, by segmenting a spectral line into segments of variable length that can be further described at level-1 by acceptable approximations of their time evolutions.

At **level-1** spectral lines are described by morphology symbols $x_k \epsilon \sum_1$ and a sequence of attributes. $\sum_1$ is an alphabet. A first-level description a_k of a line segment is expressed as follows:

$$a_k = x_k t_{bk} t_{ek} f_{bk} f_{ek} f_{Mk} f_{mk} e_{ak} \qquad (4.1)$$

where: $x_k \epsilon \sum_1$ is a morphology symbol, t_{bk} is the beginning time of the segment described by a_k, t_{ek} is the ending time, f_{bk} is the frequency of the beginning time,

k	t_{bk}	t_{ek}	f_{bk}	f_{ek}	f_{mk}	f_{Mk}	e_{ak}
1.	41	65	2754	2700	2511	2754	8.1
2.	41	49	3078	2916	2916	3078	8.3
3.	42	78	513	540	459	540	8.1
4.	42	78	540	783	540	783	7.8
5.	41	81	2160	2727	2160	2727	7.9
6.	46	57	1800	2187	1890	2187	8.1
7.	45	56	3024	3240	3024	3240	8.8
8.	51	74	2916	3375	2916	3375	8.3
9.	58	63	3267	3348	3267	3348	8.4
A.	65	81	2295	2565	2295	2565	6.8
B.	74	83	297	297	297	324	7.1
C.	74	83	2916	3213	2916	3213	5.6

Figure 4.5: Level-1 description of the pattern of Figure 4.3

f_{ek} is the frequency of the ending time, f_{Mk} is the maximum frequency, f_{mk} is the minimum frequency, and e_{ak} is the average energy.

$\sum_1$ is an alphabet obtained by concatenating two symbols belonging to alphabets $\sum_{1a}$ and $\sum_{1b}$. $\sum_{1a}$ describes temporal events and is defined as follows:

$$\sum_{1a} : \{A:ascendent, H:horizontal, D:descendent\} \tag{4.2}$$

$\sum_{1b}$ gives a rough indication of the frequency location of the mid-point of the line:

$$\sum_{1b} : \{LO:low,\ LA:low-average,\ A:average,\ AH:average-high,\ HI:high,\ VH:very-high\} \tag{4.3}$$

If a line requires more than one symbol of $\sum_{1a}$ or $\sum_{1b}$ in order to be properly described then it is automatically segmented into lines each one of which can be described by a single symbol.

Figure 4.5 shows a description of spectral lines represented in Figure 4.3.

4.1.3 - A CONTINUOUS PARAMETER AND FREQUENCY DOMAIN BASED MARKOV MODEL FOR DESCRIBING FREQUENCY RELATIONS

Frequency relations among Spectral Lines (SL) can be expressed in many ways. A particularly interesting set of descriptors is the class of **Places of**

Articulation (PA) defined by the following vocabulary:

$$\sum_{PA} : \{FP: front-place,\ CP: central-place,\ BP: back-place\} \qquad (4.4)$$

The symbols of $\sum_{PA}$ are used for **Level-2** descriptions. The literature on Acoustics-Phonetics is rich in work relating place of articulation to spectral morphologies [83]. From this knowledge we can expect different relations between SL's and PA's depending on the nature of speech segments. For some sounds, like plosives, relations involve SL transitions; for some other sounds, like non-nasalized sonorants, interesting relations can be established between PA's and spectral lines that are quasi-stationary in time. The inference of the latter type of relations will be discussed in this section.

Any speech interval containing only horizontal lines can be assumed to be quasi-stationary. The same assumption can be made with other intervals obtained by segmenting larger segments into smaller ones in which line-parameter variations are modest. Large portions of a speech signal can be characterized in terms of quasi-stationary intervals of **variable length**. These intervals can be further segmented in order to obtain **fixed-length** intervals each one of which can be described by PA hypotheses using relations with SL.

Place of Articulation is a very useful, although often not sufficient, feature for describing speech patterns. For some vocabularies, like the one consisting of letters and digits PAs are sufficient for characterizing most of the vowels and

diphthongs. Different speakers produce **different** spectral lines for the **same** PA, but such variations have **constraints** that can be expressed statistically on distortions, insertions and deletions of spectral lines.

In order to obtain more adequate descriptions of speech patterns, other features have to be considered. They can be described by other relations with spectral morphologies. In this way each speech interval can be represented by a **composite description.**

In this section the possibility of using spectral lines for the recognition of the manner of articulation together with the place of articulation will also be considered. This will make it possible to generate hypotheses about all the vowels.

A speech unit or a word can be related to chains of composite descriptions through Stochastic Automata or Markov Models. The important property of composite descriptions is that they represent clusters of acoustic morphologies whose importance is motivated by Speech Science and whose distortions can be represented by a statistical model of multispeaker performances.

4.1.4 - PARAMETER CHARACTERIZATION AND STRUCTURE OF STATISTICAL RELATIONS BETWEEN SL'S AND PA'S

Segments corresponding to vowels extracted from the pronunciation of connected letters and numbers have been used.

Places of articulation of vowels were assigned to segments using a semiautomatic procedure where the intervention of a supervisor was required only for labeling or accepting labels of difficult cases.

For each labeled interval, each spectral line was represented by two parameters corresponding to its **frequency** and its associated **spectral energy**.

In order to introduce a sort of normalization, rather than using frequencies and energies, differences between the frequency and energy of each line and the frequency and energy of a base-line are used. The base-line is the line of highest energy in the low frequency range.

The description of a quasi-stationary interval is a string Y_i of vectors of the form:

$$Y_i = y_{i1}, y_{i2}, \ldots, y_{in} \tag{4.5}$$

Each vector y_{ij} of the sequence Y_i represents a line in the pattern. The first vector y_{i1} corresponds to the base line. The remaining lines of the pattern are

Bi1	Bi2
633	8.26
-422	-0.31
-343	-0.42
486	0.25
852	0.06
1840	0.46
1979	0.87
2535	0.30
2593	0.25
2707	0.43

Figure 4.6: A quasi-stationary speech pattern and its sequence of vectors

sorted by frequency.

Each vector has two components defined as follows:

$$y_{i1}=B_{i11},B_{i12}, \cdots , y_{ij}=B_{ij1},B_{ij2}, \cdots , y_{in}=B_{in1},B_{in2} \qquad (4.6)$$

where:

B_{i11} = frequency of the base line,

B_{i12} = energy of the base line,

$B_{ij1}=f_{ij}-B_{i11}$,

$B_{ij2}=e_{ij}-B_{i12}$,

f_{ij} = average frequency of the j-th sorted line in the pattern i,

e_{ij} = average energy of the j-th sorted line in the pattern i.

Figure 4.6 shows a speech interval with the corresponding vector Y_i as defined by (4.5) and (4.6).

4.2 - FREQUENCY DOMAIN BASED MARKOV MODELS

In the next sections the theoretical definition of frequency domain based Markov models will be presented. The results of the experiments on the use of such models for speech recognition will be presented in section 6.1 and section

6.3.

4.2.1 - INTRODUCTION TO MARKOV MODELS

The theory of Markov models and their application to speech recognition is satisfactorily covered in the literature. Basic references for the elementary aspects of the theory are [41], [72], [71], [8]. Several variation and enhancement are proposed to the basic models in [2], [26], [28], [44], [88]. In the following sections a discussion of the mathematical aspects of the theory that is relevant to our approach is provided. Further details can be found in the references just cited.

Markov models are a special case of the EPN model. A state in the Markov model corresponds to a state in the network. the *a priori* probability of Markov model transition is the measure of the corresponding EPN transition. The EPN conditions are set to true and the probability distribution of Markov model transition is represented by the EPN action which returns the same probability value. Stochastic algorithms on Markov models like Viterbi algorithm and Forward-Backward algorithm are performed by the EPN supervisor with forward and backward strategy.

4.2.2 - MARKOV MODELS IN THE FREQUENCY DOMAIN

A Markov source is introduced to model a process that generates spectral lines and their energies. The model includes formants, spurious lines and lines corresponding to a split of a format into two lines. Frequency and amplitude distributions are associated with each transition in the model. The model is conceived in such a way that variances of the distributions are kept small so that each distribution represents variation due to inter-speaker differences of the parameters of a line having specific structural properties.

Distortions of frequency and energy differences are assumed to have normal distribution and to be statistically independent. A model without such simplifying assumptions would have been more realistic, but it would have implied practical complications. We decided to avoid them and to build a manageable model to be eventually compared in the future with more complex ones.

The statistical relations between SLs and PA's are characterized by a CPMM (Continuous Parameters Markov Model). A CPMM is a Markov Model in which transitions produce vectors of parameters. The probability $p(s_n,s_k)$ is the probability of choosing the transition from state s_n to state s_k when the state s_n is reached. $q(s_n,s_k,y_i)$ is the probability that the vector $y_{ij}=B_{i1},B_{i2}$ is produced in the transition from s_n to s_k. The collection of the probability distributions of the parameters describes a transition.

A transition t from s_n to s_k that produces the vector y_i is then described by the

following matrix:

$$M_t = \begin{bmatrix} m_{i1} & \sigma_{i1} \\ m_{i2} & \sigma_{i2} \\ \cdot & \\ m_{il} & r_{il} \\ \cdot & \\ \cdot & \\ m_{iL} & \sigma_{iL} \end{bmatrix} \tag{4.7}$$

where m_{il} is the mean and σ_{il} is the variance of parameter B_{il}. In our case, l can be either 1 (frequency) or 2 (amplitude).

4.3 - LEARNING AND RECOGNITION

The Forward-Backward [4] algorithm has been used for both learning and recognition purposes. On each transition the lines are assumed to be produced by a multivariate Gaussian distribution $G(t,y_{ij})$ with diagonal covariance matrix. In our case the distribution is two dimensional as we consider the probability of a line y_{ij} as a function of its parameters B_{ij1}, B_{ij2} (see (4.6)). The formula becomes:

$$G(t,y_{ij}) = \frac{1}{2\pi\sigma_1(t)\,\sigma_2(t)}\, e^{-\frac{1}{2}\left[\frac{B_{1ij}-m_1(t)}{\sigma_1(t)} + \frac{B_{2ij}-m_2(t)}{\sigma_2(t)}\right]^2} \tag{4.8}$$

where $m_1(t)$ and $m_2(t)$ are the means of line parameters B_{ij1}, B_{ij2} and $\sigma_1(t)$ and $\sigma_2(t)$ are the standard deviations of the same parameters. The distribution parameters m and σ have to be estimated through the learning set $A = Y_1, Y_2, \cdots, Y_m$ starting from an initial hypothesis. Let us define $p(t,y_{ij})$ as the joint probability that Y_i is the pattern produced by a HMM and the j-th line whose parameters are represented by the vector y_{ij} is produced by the transition t. Formally $p(t,y_{ij})$ is defined as:

$$p_i(t,y_{ij}) = \begin{cases} \alpha_{ij}(q_s)p_t G(t,y_{ij})\beta_{i+1,j}(q_e) & \text{if } t \textit{ produces outputs} \\ \alpha_{ij}(q_s)p_t\beta_{ij}(q_e) & \text{if } t \textit{ produces the null output} \end{cases} \tag{4.9}$$

where q_s is the startpoint of t, q_e is the endpoint of t, p_t is the *a priori* probability of the transition t, $\alpha_{ij}(q)$ is the forward probability of state q at instant i while the output Y_i is produced, and $\beta_{ij}(q)$ is the backward probability of state q at instant i while the output Y_i is generated by the Markov model. The reestimation formulas that have been used to estimate the mean and the standard deviation of line parameter k, and the *a priori* probability of a transition t follow. The mean is:

$$m_k(t) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{B_{ijk}p_i(t,y_{ij})}{p(Y_i)}}{\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{p_i(t,y_{ij})}{p(Y_i)}} \tag{4.10}$$

In (4.10) $p(Y_i)$ can be factored and eliminated, therefore obtaining:

$$m_k(t) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} B_{ijk}p_i(t,y_{ij})}{\sum_{i=1}^{m}\sum_{j=1}^{n} p_i(t,y_{ij})} \tag{4.11}$$

The standard deviation is:

$$\sigma_k(t) = \left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} (B_{ijk} - m_k(t))^2 p_i(t,y_{ij})}{\sum_{i=1}^{m}\sum_{j=1}^{n} p_i(t,y_{ij})} \right)^{\frac{1}{2}} \tag{4.12}$$

upon factorization and elimination of $p(Y_i)$ as in (4.10). For computational efficiency, even though it is less accurate due to round-off errors, formula (4.12) can be re-written as:

$$\sigma_k(t) = \left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} B_{ijk}^2 p_i(t,y_{ij})}{\sum_{i=1}^{m}\sum_{j=1}^{n} p_i(t,y_{ij})} - 2m_k(t) \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} B_{ijk} p_i(t,y_{ij})}{\sum_{i=1}^{m}\sum_{j=1}^{n} p_i(t,y_{ij})} - m_k(t)^2 \right)^{\frac{1}{2}} \tag{4.13}$$

The estimated frequency of transition t is:

$$f_t = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_i(t,y_{ij})/p(Y_i)}{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{t'/L(t)=L(t')} p_i(t',y_{ij})/p(Y_i)} \qquad (4.14)$$

where the function L(t) returns the startpoint of the transition t.

During learning and recognition a scaling technique similar to the one described in [54] has been adopted.

In the recognition process, the probability $p(Y_i/M_k)$ is computed with the Forward-Backward algorithm as follows:

$$p(Y_t) = \sum_{q\in Q} \alpha_{nt}(q) \qquad (4.15)$$

where Y_t is an input string of vectors as defined by (4.5) and M_k is a CPMM.

The string Y_t is assigned to the j-th Place of Articulation PA_j if

$$p(Y_t/M_j) = \max_{h=1}^{H} \{p(Y_i/M_h)\} \qquad (4.16)$$

where H = 3 for VB, VC, and VF, and:

$$(p(Y_t/M_j) - p(Y_t/M_k)) > c_{jk} \qquad (4.17)$$

where c_{jk} is the threshold of confusion between M_j and M_k (the Markov source corresponding to the second highest score).

If

$$(p(Y_i/M_j) - p(Y_i/M_k)) \leq c_{jk} \qquad (4.18)$$

then PA is decided (if a local decision has to be made) according to rules.

In a program for the automatic recognition of PA for vowels, the following rules can be used in order to decide when probabilities for "back" and "front" places of articulation are very close:

1) if $g_1 > g_2$ then "PA: back"

2) if $g_1 < g_2$ then "PA: front"

where:

$$g_1 = \max_{i=1}^{n} \{B_{ij2} | ((B_{ij1} > B_{i11}) \text{ and } (B_{ij1} < th_1))\}$$

$$g_2 = \max_{i=1}^{n} \{B_{ij2} | ((B_{ij1} > th_1) \text{and} (B_{ij1} < th_2))\}$$

th_1 = 1500 Hz; th_2 = 2900 Hz.

4.4 - DIPHTHONG OPERATOR

The speech signal is segmented to produce segments showing a quasi stationary pattern of lines. The smallest segment is of course a single frame. Let $S_i = s_{i1}, s_{i2}, \cdots, s_{in}$ be the sequence of segments that produce pattern $Y_i = y_{i1}, y_{i2}, \cdots, y_{in}$. The probability vector $y_{ij} = \mu_{ij1}, \mu_{ij2}, \cdots, \mu_{ijH}$, where $\mu_{ijk} = p(y_{ij}/M_k)$ is the probability that pattern Y_i be generated by model M_k. In our case $H = 3$ and the three models recognize the places of articulation

back, central, and front.

A diphthong is represented by a Markov model D_x in the time domain, whose transitions t are described by the 4-tuple $(p_t, s_n, s_k, M(t))$ where p_t is the *a priori* probability of the transition, s_n is the startpoint of t, s_k is the endpoint of t, and M(t) is the model that generates y_{ij} in pattern Y_i at instant j, with probability μ_{ijh}. All the models together constitute the set $D = D_1, D_2, \cdots, D_N$ of models of diphthongs.

The Viterbi algorithm is used to find the sequence $S^* = t_1, t_2, \cdots, t_n$ of transitions of the model D_ι that maximizes the probability

$$p(Y_i/S) = \prod_{j=1}^{n} \left[p_{t_j} \, p(y_{ij}/M(t_j)) \right]$$

It is assumed that $p(Y_i/D_\iota) = p(Y_i/S^*)$. Pattern Y_i is assigned to diphthong class $D^* = \underset{D_\iota \in D}{argmax}\, p(Y_i/D_\iota)$. Constraints on durations can be inserted in the model following the basic scheme in Figure 4.7, in which strings of length greater than m can be assigned a low probability (remark the transitions leaving from S_m), and strings of length less than n are not accepted. It is important to remark that no learning is required for such models: all the learning is concentrated in the models in the frequency domain.

The results and the experimental details of the application of the diphthong operator will be presented in section 6.2.

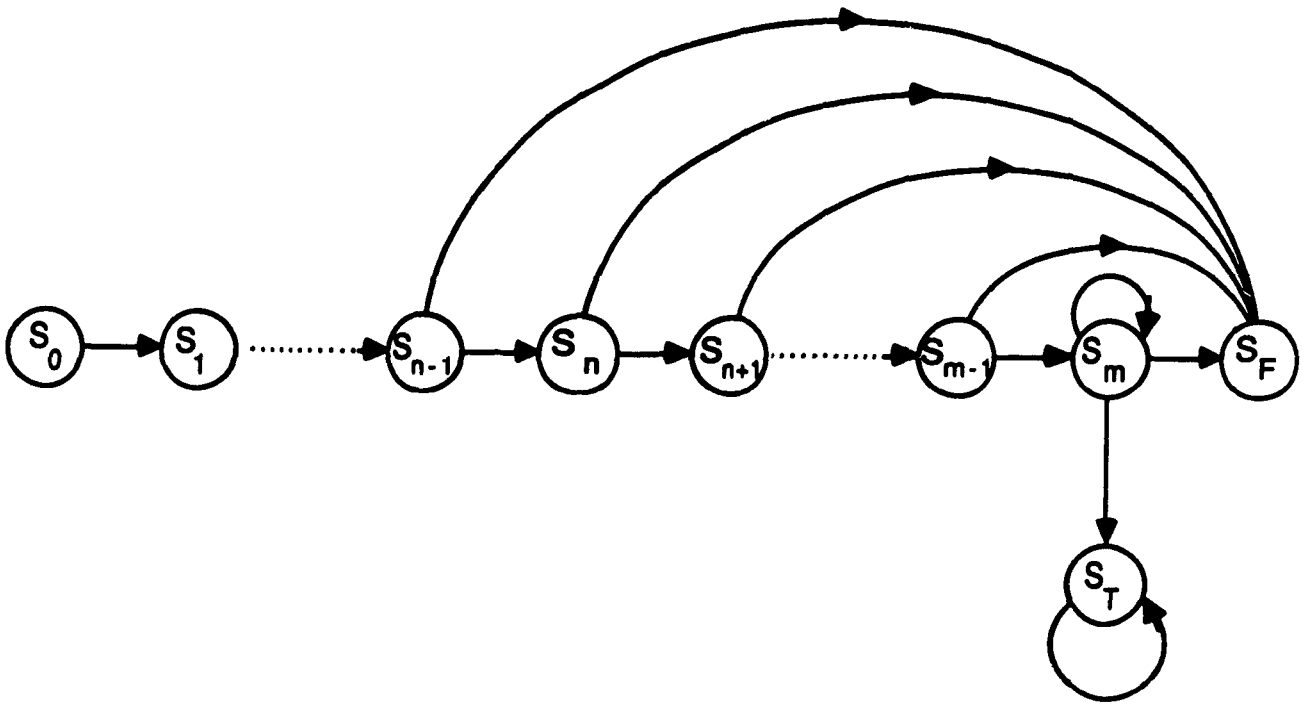


Figure 4.7: Markov submodel for time constraints

4.5 - NEURAL NETS OPERATOR

The pattern Y_i corresponding to the speech signal is analyzed frame by frame by different sets of neural nets [14]. Let

$$M_{j1} = \{\mu_{j11}, \mu_{j12}, \cdots, \mu_{j1n_1}\}$$

$$M_{j2} = \{\mu_{j21}, \mu_{j22}, \cdots, \mu_{j2n_2}\}$$

$$\cdots \quad \cdots \quad \cdots \quad \cdots$$

$$M_{jm} = \{\mu_{jm1}, \mu_{jm2}, \cdots, \mu_{jmn_m}\}$$

be the sets of memberships calculated on the j-th frame by MLNs M_1 to M_m.

A word is represented by a pseudo Markov model W_x in the time domain, whose transitions t are described by the 3-tuple (s_n, s_k, f_t) where s_n is the startpoint of t, s_k is the endpoint of t, and f_t is a function associated to the transition t. Function $f_t : M_{j1} \times M_{j2} \times \cdots \times M_{jm} \rightarrow [0,1]$ returns a membership which is a function of input memberships μ_{tjk}. The set of all the words is $W = W_1, W_2, \cdots, W_N$.

The Viterbi algorithm is used to find the sequence $S^* = t_1, t_2, \cdots, t_n$ of transitions of the model W_x that maximizes the membership

$$\mu(Y_i/S) = \sum_{j=1}^{n} f_{t_j}(\mu_{j11}, \cdots, \mu_{jmn_m})$$

We define $\mu(Y_i/W_x) = \mu(Y_i/S^*)$. Pattern Y_i is assigned to word class $W^* = \underset{W_x \in W}{argmax}\ \mu(Y_i/W_x)$. Constraints on durations can be inserted in the model

following the basic scheme in Figure 4.7, in which strings of length greater than m can be assigned a low probability (remark the transitions leaving from S_m), and strings of length less than n are not accepted. It is important to remark that no learning is required for such models: all the learning is concentrated in the MLNs.

The experimental details and results of the application of the neural net operator will be presented in section 6.6.

4.6 - CONCLUSIONS

New and highly specialized operators have been defined to perform specific analyses of the speech signal. The EPN paradigm has the flexibility of integrating these heterogeneous operators into strategies for ASR. Next chapter will introduce strategies for different ASR experiments.

CHAPTER - 5

5 - ASR RECOGNITION STRATEGIES

This chapter presents several strategies that make use of the operators of chapter 4. Different problems are attacked. Section 5.1 shows the approach to the recognition of letters and digits. Section 5.2 presents the strategy for lexical access and section 5.3 introduces a strategy for speech recognition that makes use of neural networks.

5.1 - PROCEDURAL NETWORKS FOR LETTERS AND DIGITS RECOGNITION

The EPN model has a **data-driven** component that identifies **Acoustic Segments** (AS) based only on acoustic evidence and knowledge of the information bearing properties corresponding to different spectral structures. An Acoustic Segment usually contains at least a "vocalic" part identified by resonances represented by narrow band spectral lines in a time-frequency-energy representation of speech. An AS may contain one or more vowels with one or more consonants in the "vocalic" part. An AS may also have a **head** and a **tail.** Heads and tails may contain low energy sonorant consonants or consonants characterized by frication noise or by a transition between a deep dip in the signal energy curve and an energy peak.

The coarse acoustic properties that characterize ASs and are used for delimiting them are by no means **interpretations**; they are just elements for

focusing the attention of the property extractors. Relations between ASs and SUs like phonemes, diphones or syllables are established by **performance models** representing, for each SU of interest, insertions, deletions, substitutions and their statistics.

As unambiguous segmentation of continuous speech into ASs is very useful for reducing the complexity of word hypotheses generation and verification because word hypotheses can start only at specific time instants of the head, vocalic part or tail of an AS.

For the head, vocalic part and tail of each AS, plans of property extraction **operators** (procedures) are executed. These plans produce descriptions of speech segments. These descriptions may apply to segments of **variable duration.** Ferguson [31] has shown how performance models can be build under the assumption that properties generated by a model have variable duration. Plans producing descriptions perform a sort of **Variable Depth Analysis** (VDA) because they may generate different types of properties for different segments.

The results of VDA can be **descriptors** obtained by a sort of "Knowledge-based vector quantizer". They can also be Speech Unit hypotheses affected by a certain score. These scores that are generated by procedures are used by performance models consisting in stochastic automata relating words and SUs.

Unfortunately there is not a suitable theory for the conception of performance models in such a case, but interesting research is in progress [36]. Such research is motivated by the existence of a similar problem in different

application areas [3], [80]. While waiting for a probabilistic theory of heterogeneous pattern descriptions, a pseudo solution can be adopted where word hypotheses scores are obtained by just multiplying probabilities or scores of each segment even if the type of acoustic properties may vary from segment to segment.

Most of the operators associated with EPN actions include plans, Hidden-Markov-Models (HMM), local parsers, rule-based inference units. These actions produce scored interpretations of segments of the speech signal. All these tools are used for extracting an unambiguous description D of a speech pattern and for computing an a-priori probability for an hypothesis H:

$$P(D/H) \tag{5.1}$$

The *EPN* supervisor keeps up-to-date a search space where each node is represented by the following four-tuple:

$$(q_{ei}, q_{bi}, context, score) \tag{5.2}$$

where:

- q_{ei} and q_{bi} are respectively the endpoint and the startpoint of the transition
- "context" represents the contextual information associated to q_{ei} It contains the time interval of the speech signal in which the execution of sensory procedures invoked by the transition is performed. Segmental information and the current hypothesis are also included in the context.
-"score" is the score of the hypothesis contained in or implied by "context" in the specified time interval; score could be $P(D(t_\phi T)/H)$ where t_ϕ is the beginning of

the sentence. T could also be a set of possible time references; in this case, score will be a set of scores $\{s(t)/t \in T\}$. Composite scores can be evaluated as likelihoods:

$$L(D,H)=Pr(D/H)Pr(H) \tag{5.3}$$

where $Pr(H)$ is obtained by a language model.

Global information is also accessible by the EPN supervisor at any time. Markov models, linguistic description of the signal, power spectrum and so forth are kept in the global communication area.

The size of the search space can be kept small in spite of a large number of states in the *EPN* if conditions and actions are properly chosen and placed in the network.

The assumption made for the experiment described in this work is summarized in the following. Score p_i is the a priori probability of an arc; g_i is the probability that the condition is satisfied; h_i is the probability that the segment s_k matches with the knowledge used by action act_i, f_i is a multiplication operator, The contribution (2.3) can be rewritten as:

$$u_i=p_i g_i(cond_i)h_i(act_i) \tag{5.4}$$

Let $s=s_1,s_2,\ldots,s_n$ be an input sequence of speech segments, $EPN_k=\{k,Q,A,q_o,q_f\}$ be the k-th EPN and $a=a_1,a_2,\ldots,a_n$ be a sequence of arcs in the network EPN_k such that the initial state of a_1 is q_ϕ and the terminal state of a_n is q_f.

The supervisor attempts to find the sequence of arcs "a" which maximizes the conditional probability

$$P(a/s)=P(s/a)/P(s) \tag{5.5}$$

for a given s, that is, to find the sequence "a" which maximizes:

$$P(s/a)=\prod_{i=1}^{n}\left[P_i g_i(cond_i)h_i(act_i)\right] \tag{5.6}$$

An example of an EPN to be used to compute the score of the word /five/ is shown in Figure 5.1. The EPN is supposed to extract data from the AS under analysis and to produce a score that is an estimation of the a-priori probability that the data extracted from AS have been observed during the pronunciation of /five/.

The initial state is associated with a context containing the hypothesis /five/. The first arc is associated with a PUSH action whose function consists in using a subnetwork for generating hypotheses about fricative sounds on the head of a segment. Action PUSH head (fr) extracts the head of the AS under analysis and executes a "Network of Actions" on the segment head for computing the acoustic properties that are relevant for discriminating among fricative sounds. Let $data_1$ be the properties extracted from the AS head. Properties $data_1$ are related to the SU /f/ and the following score is computed:

$$Pr(data_1/f)$$

The above probability could be obtained directly or through the probabilities of the **place** and **manner of articulation** for f.

In the case of the following diphthong, the SU /ai/, there is no need to hypothesize the manner of articulation in order to distinguish it from the other syllables of the language to be recognized. Thus, only the properties $data_2$ for the place of articulation are considered and the score (vf stays for front-vowel, vc stays for central vowel):

$$Pr(data_2/vc \quad vf)$$

are computed by the action **PUSH vocalic.** An algorithm for the computation of the above probability using HMM in the frequency domain is proposed in [62]. In a similar way a "tail (fr)" network is invoked for generating hypotheses about fricatives in the tail of the segment. It will extract data, and compute a score for /v/.

Eventually, the final state 53 is reached and the action POPABS f is executed. The associated function f for computing the cumulative score returns $Pr(data/5)$, where $Pr(data/5) = Pr(data/f)\ Pr(data_2/vc \quad vf)\ Pr(data_3/v)$.

Networks like the one shown in Figure 5.1 are examples of a **model driven** approach to word hypotheses. In such an approach there is a procedural model for each hypothesis that can be generated. The model driven approach and the example of Figure 5.1 have some problems that will be discussed in the following.

The first problem is that EPNs like the one of /5/ and the one of /9/ have a lot of actions in common whose execution should not be duplicated. A more efficient network organization will be presented in section 6.4.

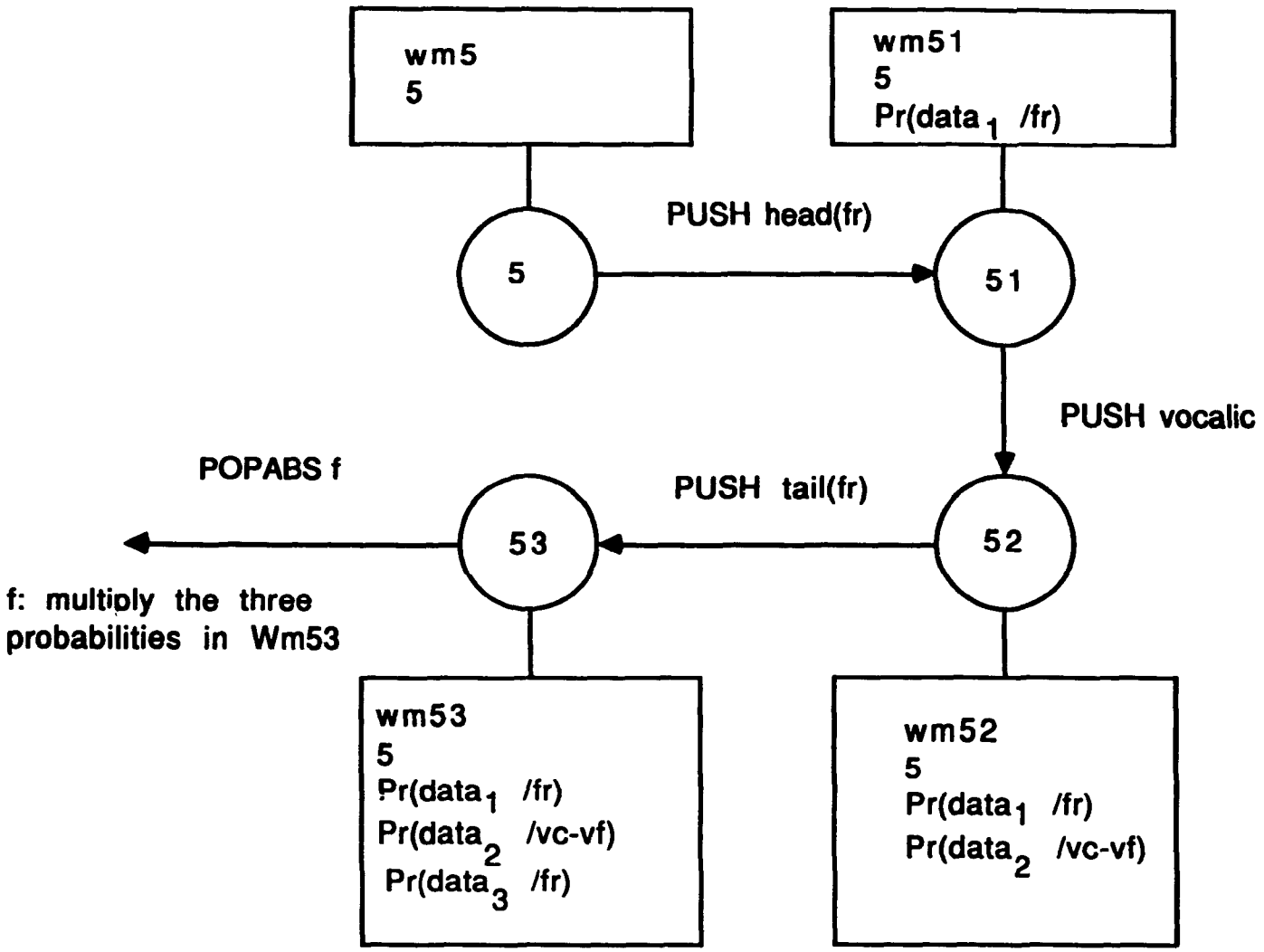


Figure 5.1: Example of an Extended Procedural Network (EPN)

HIM	=	NC V SON
BEEN	=	NI V SON
IF	=	V NC
MOST	=	SON V NC NI
MUST	=	SON V NC NI
OUT	=	V SON NI
SO	=	NC V
WHAT	=	SON V NI

Figure 5.2: Subset of vocabulary

Table 5.1: Basic PPF classes

NC
NI
SON
V
WF

Table 5.2: PPF based on coarticulatory effects

SV (SON-V)
SVS (SON-V-SON)
VS (V-SON)
VSV (V-SON-V)
VV (V-V)
WV (WF-V)
NIF (final NI)
NIS (NI-SON)
SVSV (SON-V-SON-V)
VSVS (V-SON-V-SON)
SW (SON-WF)
VW (V-WF)
NCF (final NC)
NCS (NC-SON)

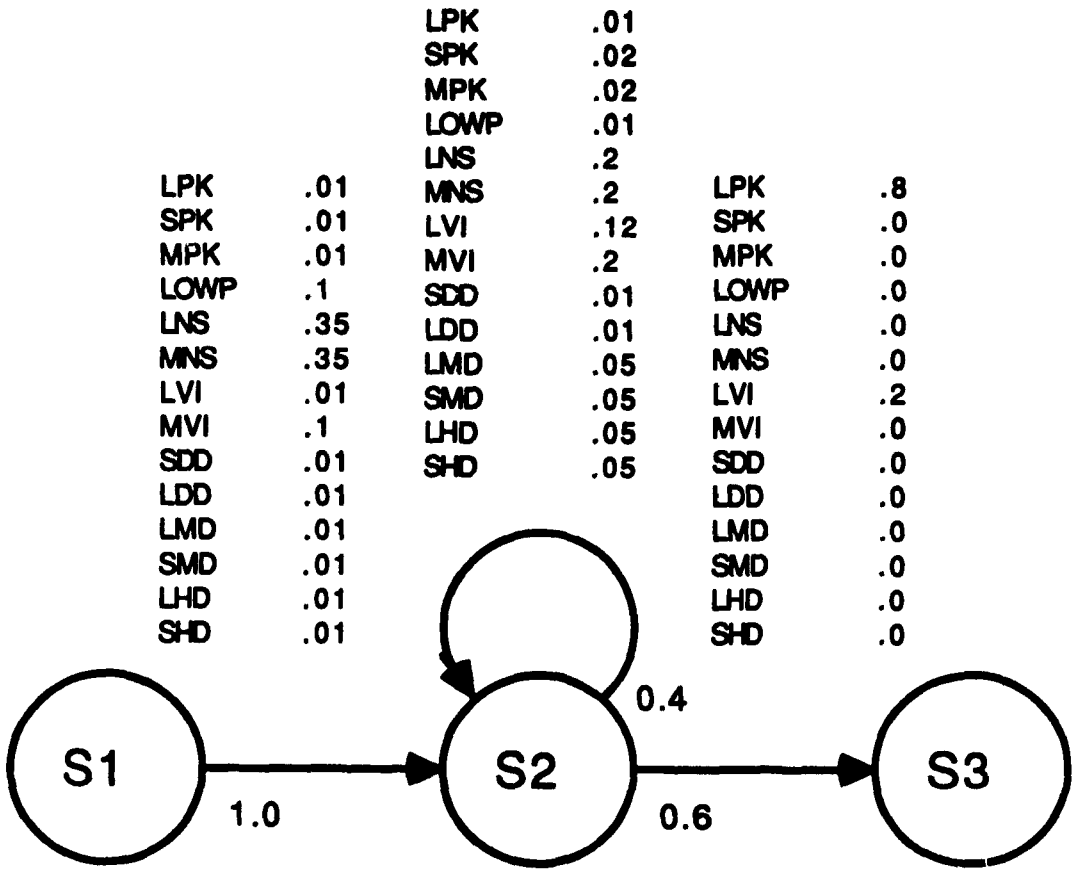


Figure 5.3: Example of Markov model for "fricative-vocalic" (fr-vw)

The second problem is that different networks may extract different types of data in the same AS making score comparisons rather difficult.

This problem can be avoided by organizing the EPN hierarchy in such a way that properties for each interval of an AS are extracted only by a single action.

The third problem is that properties like $data_1$, $data_2$ and $data_3$ extracted in different intervals of the same AS may be different in number and in quality. This problem is common to other Pattern Recognition applications [3], [36], [80]. A pseudo-solution of such a problem has been adopted for the project described in this paper consisting in just multiplying probabilities of different segments. Other solutions are under investigation.

A fourth problem is that the boundary between heads, vocalic parts and tails may be fuzzy. This possibility is not considered in the application described in this research although it will be investigated in future works.

Section 6.4 will report the experimental results and set-up for the recognition of letters and digits.

5.2 - LEXICAL ACCESS

5.2.1 - INTRODUCTION TO LEXICAL ACCESS

The ultimate goal will only be reached when Speech Recognition Systems will be speaker-independent and will deal with vocabularies that approach in size

those typically commanded by human beings. A human being can usually recognize from 50,000 up to 100,000 words, although he may have a conscious knowledge of only a fraction of this amount [84]. However, to converse with one another people normally use a maximum of 12,000 words [46]. Therefore a vocabulary of this size would suffice to make the interface between man and machine approach the reality of vocal communication between human beings.

Systems presently in general use deal with very limited vocabularies (less than 1000 words) and are directed at very specific and specialized tasks; moreover, since most are speaker-dependent they need a special training from the would-be user. This training is lengthy and delicate in spite of the reduced vocabulary size involved.

Interesting laboratory prototypes have been developed for the recognition of 10 to 20 thousand words [35] [43], [50], [60]. Such systems are speaker-dependent and most do not recognize connected speech.

Systems that rely upon template-matching techniques are hardly manageable if extended to the size of large lexicons. The complexity of such an approach depends on the number of templates. The reduction of a smaller recognition unit (for example from words to syllables, doesn't help; in English there are about 20,000 syllables [67].

The main problem with large vocabularies still remains the high degree of confusion inherent in them. In a study using the original phonetic labeling from the Webster dictionary, close to 30,000 minimal pairs of words have been found

[84]. Furthermore, a statistical study of spoken English [25] reported that in most minimal pairs, the distinguishing phonemes differ by their manner of articulation rather than their place of articulation, thus aggravating the confusion. This suggests that very fine acoustic information has to be extracted from the signal in order to correctly recognize phonetic segments.

Phonetic recognition, as opposed to template-matching, has the advantage of exploiting phonemic constraints and distributions in large lexicons. Shipman and Zue [81] showed that a large vocabulary can be reduced to a very small list of candidates (on the average less than one percent of the total vocabulary) if the utterance is described in terms of six coarse phonetic classes. Search space is thus drastically reduced and costlier methods can then be used to examine in details the diminished list of candidates.

Other Approaches in Word Recognition include: the CMU Continuous Speech Recognition System [76], the Torino Large Lexicon Access Task System [15], [50], the IBM-France Very Large Size Dictionary Speech Recognition System [60], the BNR Large Vocabulary Word Recognizer [35], the NEC Large Vocabulary Word Detection System [37], the Tangora 20,000 Word Speech Recognizer [43], and others [40].

5.2.2 - EPN FOR LEXICAL ACCESS

The EPN is a valuable tool to describe and implement the access procedure to a large lexicon. Lexicons may have different representation, but the EPN can

support the most general one: graph representation. A lexicon that is organized as a tree is a particular case of a graph. EPN is hierarchical, so the lexicons can be described using different details of description at different level of representation. The built-in network actions add power and generality to the recognition process that has been designed on the lexicon. User-defined conditions and actions allow the definition of a very flexible recognition environment.

Generally speaking the lexical access is not performed on the base of the whole words, but smaller Speech Units (SU) are used. Different approaches make use of different units. In our case the units are called PPF (Primary Phonetic Features) [20]. The basic five PPF classes are reported in Table 5.1.

All the words in the dictionary are described according to the chosen strategy. A subset of our dictionary is shown in Figure 5.2.

Several words may have the same PPF description. Conversely, a set of words is attached to a specific PPF description. This subset is called a cohort. Therefore the result of the lexical access is a set of words that correspond to the PPF description recognized in the signal.

The relation between PPFs and signal is described by HMMs. Each PPF is represented by a three-state Markov Model whose transitions are taken according to the transition PAC distribution (Figure 5.3). PACs (Primary Acoustic Cues) [21] are extracted from the signal. Each HMM has been learnt in a speaker independent way [57]. The word model for recognition purpose are then obtained by concatenating the HMM corresponding to the PPF description of the word

Figure 5.4. Unfortunately, this approach is far too simple because of coarticulatory effects, which are contextual modifications of phonemes. To better represent the real world several PPF classes that take into account the coarticulatory effect have been defined (see Table 5.2). Several advantages have risen from the use of EPN. The coarticulatory effects can be represented in a descriptive way (an arc on a graph). Furthermore, subnetworks give rise to a highly parallel architecture. As example, the subnetwork SONV is subsequently decomposed into three subnetworks: the proper SONV model in parallel with the concatenation on the SON and V models (Figure 5.5).

A special user defined action PUSHSYM has been defined for the purpose of implementing the concatenation of markov models. The action PUSHSYM m, where m is a HMM, receives a set of scored and time-stamped initial states

$$S = (s_1', t_1'), (s_2', t_2'), \cdots, (s_n', t_n')$$

as input and returns a set of scored and time stamped final states

$$F=(s_{1,1}'', t_{1,1}''), (s_{2,1}'', t_{2,1}''), \cdots, (s_{m,1}'', t_{m,1}''), \cdots, (s_{m,n}'', t_{m,n}'')$$

as output. The score attached to states represents the probability of being in that state at the stamped time. The score of each initial state s_i', rather than the certain probability, is successively attributed to the initial state of the HMM m. This initial state is put in column t_i' of the lexical trellis. The corresponding scores appearing attributed to the final state of M at times $t_{1,i}'', t_{2,i}'', \ldots, t_{m,i}''$ during the analysis of the signal from t_1' to its end at time $t_{m,i} = t_{m,1} = t_{m,2} = \cdots = t_{m,n}$ are called $s_{1,i}'', s_{2,i}'', \cdots, s_{m,i}''$ and the pairs

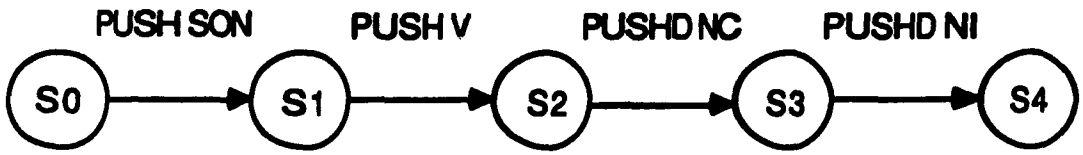


Figure 5.4: EPN corresponding to the word "MOST"

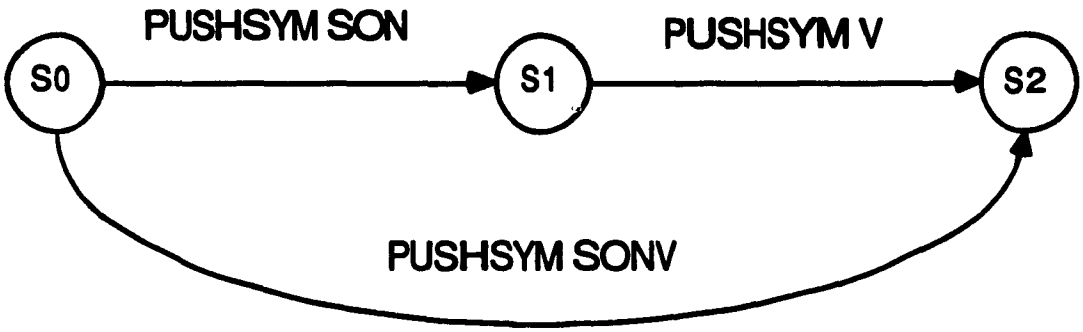


Figure 5.5: EPN corresponding to the PPF SONV

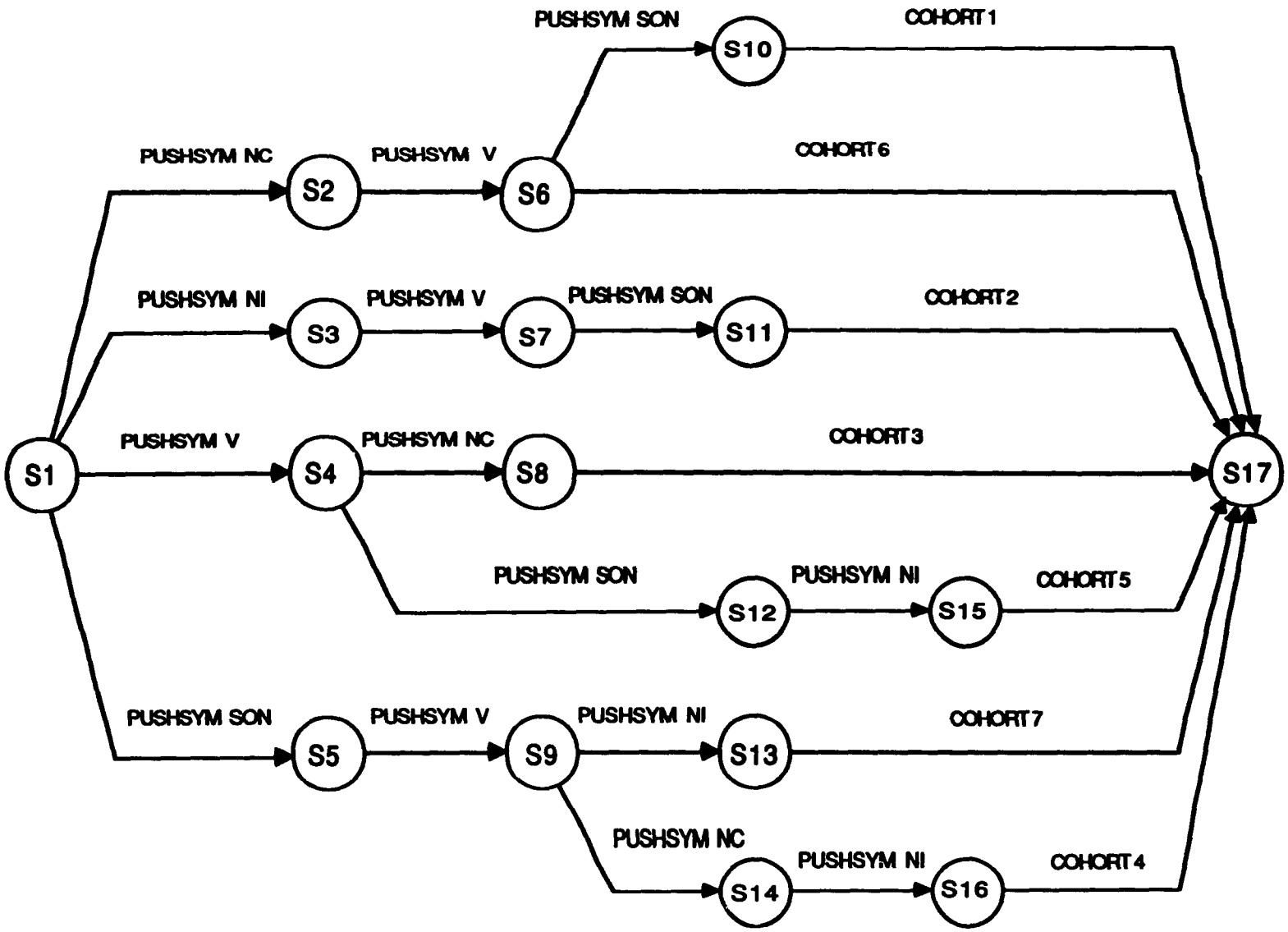


Figure 5.6: EPN corresponding to the lexicon of Figure 5.2

COHORT CODE		COHORT
1	=	NC V SON
2	=	NI V SON
3	=	V NC
4	=	SON V NC NI
5	=	V SON NI
6	=	NC V
7	=	SON V NI

Figure 5.7: Cohort list for the vocabulary of Figure 5.2

WORD	WORD CODE	COHORT CODE
HIM	1	1
BEEN	2	2
IF	3	3
MOST	4	4
MUST	5	4
OUT	6	5
SO	7	6
WHAT	8	7

Figure 5.8: Word code and cohort code for the vocabulary of Figure 5.2

formed by score and time are put in the set of scored and time stamped final states. The states and the related information are propagated through the state buffer of the EPN. The action PUSHSYM computes in an efficient way the contribution of each initial state and signal interval to the global lexical trellis. The principles of "overlapping contributions" takes place and each partial computation is halted whenever it is sure that the contribution wouldn't modify the optimal selection previously made. In this way only a small amount of average overhead is paid for the flexibility of describing the lexicon in the EPN formalism.

The EPN structure corresponding to the dictionary of Figure 5.2 is shown in Figure 5.6. Figure 5.7 and Figure 5.8 show the cohort set and dictionary with word code and cohort code.

Experimental results on the lexical access will be presented in section 6.5.

5.3 - PROGRAMMABLE EXECUTION OF MULTI-LAYERED NETWORKS

Characterizing Speech Units (SU) in terms of speech properties or speech parameters requires a form of learning with a relevant generalization capability. Structural and stochastic methods have been proposed for this purpose [21], [42]. Recently, a large number of scientists have investigated and applied learning systems based on Multi-Layered Networks (MLN). Definitions of MLNs, motivations and algorithms for their use can be found in [10], [38], [70], [77], [85], [87]. Theoretical results have shown that MLNs can perform a variety of complex

functions [77]. Furthermore, they allow competitive learning with an algorithm based on well established mathematical properties.

Our interest in the use of MLNs is justified by previously published work. We have introduced a data-driven paradigm for extracting acoustic properties from continuous speech [22] and have investigated methods based on fuzzy or stochastic performance models for relating acoustic properties with SUs. MLNs appear to be good operators for automatically learning how to extract acoustic properties and relate them with phonetic features and words automating most of the activity which formerly required a large amount of effort from a human expert. The human expert used knowledge acquired by generalizing observations of time-frequency-energy patterns. We will investigate in this section how such learning can be performed by a set of MLNs whose execution is decided by a data-driven strategy. By applying an input pattern to an MLN and clamping the output to the values corresponding to the code of the desired output, weights of connections between MLN nodes can be learned using error-back propagation [70]. When a new input is applied to an MLN, its outputs may assume values between zero and one. If we interpret each output as representing a phonetic property, then the output value can be seen as a degree of evidence with which that property has been observed in the data [17]. If phonemes are coded using a known set of phonetic features, the MLNs will learn how to detect evidence of each feature without being told all the details of the acoustic properties relevant for that feature. Statistical distributions of feature evidences can be collected in

performance models of SUs conceived as Hidden Markov Models (HMM). These models can be used to represent the time evolution of feature evidences for each SU or word. It is also possible to compute distances between time evolutions of real and desired degrees of evidences and to use such distances to rank word hypotheses, each word being characterized by a desired time evolution of degrees of evidences. Details about organization of multilayered networks can be found in [7].

5.3.1 - USE OF MLNs IN A RECOGNITION SYSTEM

The speech signal is initially segmented into two types of regions. These regions and transitions between them define situations. Each region is labeled with one of the following symbols: SON (attached to a segment with narrow band resonances - typically vowels, nasals and sonorant consonants), NS (attached to segments with a spread of energy in higher frequencies - typically fricative sounds - or to segments with a very low total energy). Each type of segment may contain every phoneme, but different sets of MLNs and MLN inputs are used for different types of segments. The system that performs a data-driven execution of MLNs is described by an Extended Procedural Network (EPN). A general purpose environment for developing various types of EPNs has been developed [22]. A simple EPN has been conceived to implement the programmed execution of MLNs.

The speech signal is considered as a sequence of segments of type SON, NS, and of transitions between segments. In Figure 5.9 the step corresponding to the analysis of a transition followed by an NS or SON segment is shown. This step is repeated as many times as the length of the signal requires. Figure 5.10 shows a two-steps variable-depth strategy associated with a segment. Depending on the preconditions, a particular set of MLNs is activated. The variable-depth paradigm is described in the following. If two or more candidates have scores which are close enough to trigger the variable-depth analysis, then an MLN specialized to solve the specific conflict is executed. If the candidates are well discriminated, the execution of specialized MLNs is not required and the default transition is taken. The number of conflict sets is finite and small. Several stages of variable-depth analysis can be considered, although, in practice, there are no more than two of them. Variable-depth analysis is particularly useful, for example, to discriminate between /m/ and /n/ when these sounds have close degrees of evidence or to discriminate among pairs of plosive sounds.

5.4 - CONCLUSIONS

The EPN paradigm has been used to define the strategy to solve some ASR problems. Such a strategy involved the use of heterogeneous operators. Next chapter will describe the experimental set-up and the results obtained in ASR applications.

Figure 5.9: Analysis of a transition followed by a segment

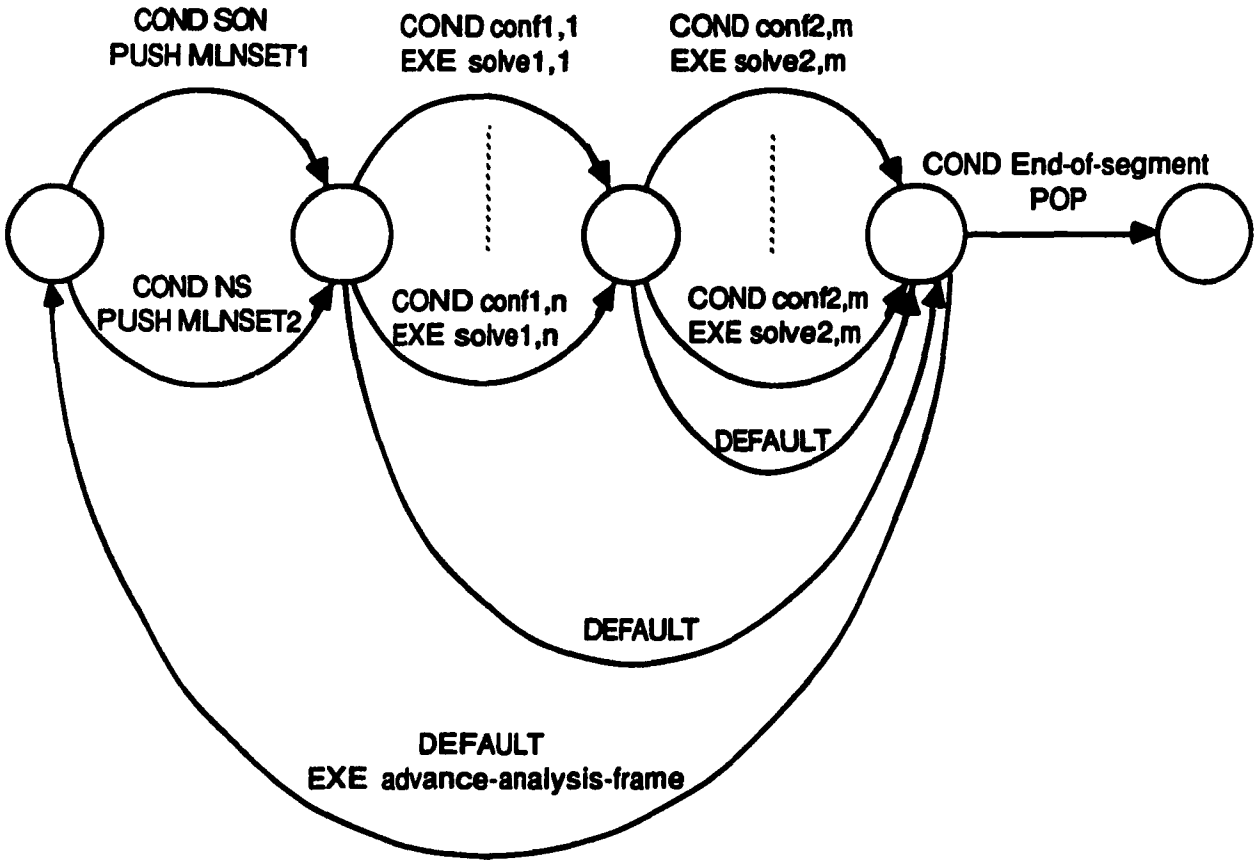


Figure 5.10: Two-steps variable-depth analysis

CHAPTER - 6

6 - EXPERIMENTAL RESULTS

Results from various experiments are reported. Section 6.1 shows the results of the recognition of the place of articulation. Section 6.2 deals with the recognition of diphthongs. Section 6.3 shows the results of vowel and diphthong recognition. Section 6.4 presents the results of letter and digit recognition. Section 6.5 deals of lexical access and section 6.6 presents the application of the neural net operator to isolated digit recognition.

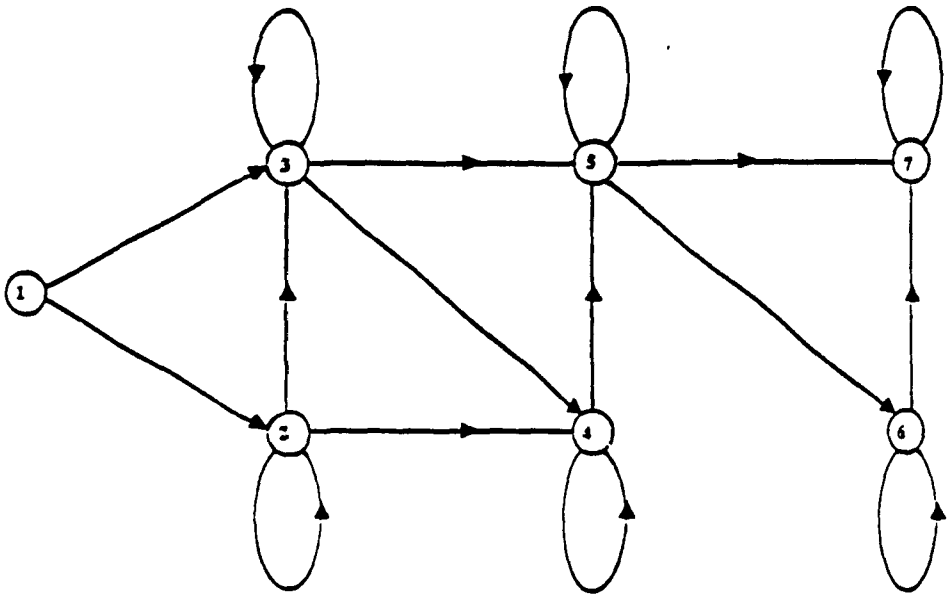
6.1 - EXPERIMENTS ON THE RECOGNITION OF THE PLACE OF ARTICULATION

This experiment was based on the operators presented in section 4.1 and section 4.2. Vocalic segments corresponding to 500 pronunciations of vowels in continuous speech from 25 female and 25 male speakers were considered for learning 3 CPMM, one for each PA in $\sum_{PA}$.

Learning was performed using the Forward-Backward algorithm on vowels extracted from random sequences of connectedly spoken letters and digits.

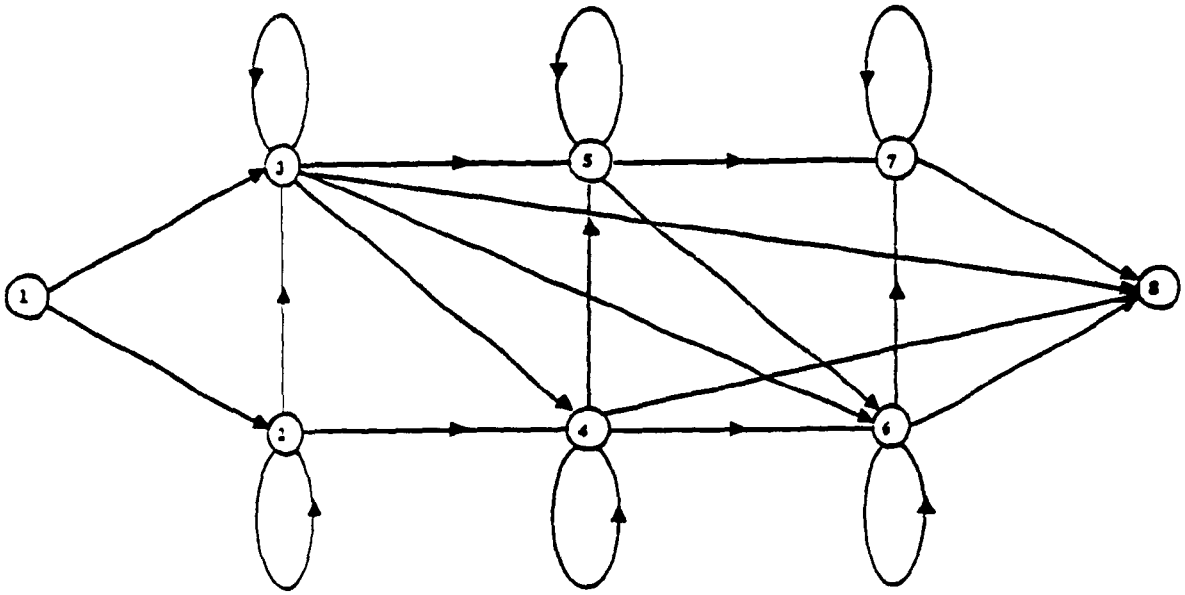
CPMM were obtained for the three places of articulation, namely FP, CP, BP.

Figure 6.1, Figure 6.2, and Figure 6.3 show the CPMMs obtained. A test set consisting of vowels pronounced by 9 new female and 9 new male speakers was used. The results are summarized in Table 6.1 and show the contribution of rules



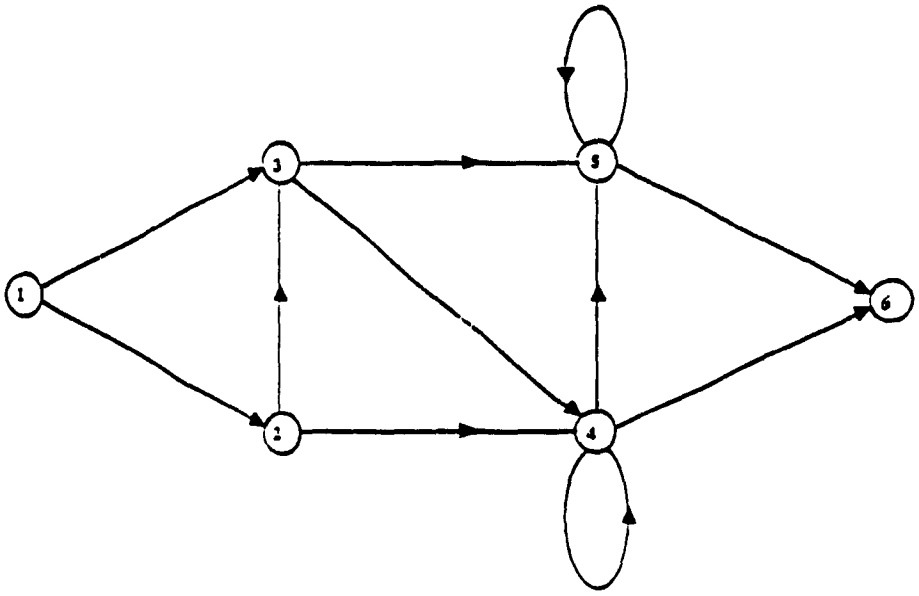
Transition		Probability of Transition	Parameter # 1		Parameter # 2	
from state	to state		Mean (M)	Variance (σ)	Mean (M)	Variance (σ)
1	3	0.438	458.790	5935.368	8.033	0.386
1	2	0.562	522.003	4019.573	8.488	0.293
2	4	0.509	311.666	6104.020	-1.726	0.091
2	3	0.327	228.448	1803.066	-1.261	0.500
2	2	0.164	-136.760	14552.347	-0.812	0.223
3	5	0.488	1777.933	57545.125	-0.016	0.498
3	4	0.297	639.481	275764.000	-1.222	0.878
3	3	0.215	444.053	6388.213	-1.835	0.191
4	5	0.704	1731.968	39209.133	-0.340	0.631
4	4	0.296	1502.419	14644.381	-0.467	0.858
5	7	0.357	2320.153	89535.836	-0.159	0.536
5	6	0.342	2052.133	4344.315	0.054	0.561
5	5	0.302	1865.370	8709.803	-0.251	0.722
6	6	0.217	2157.887	2999.774	0.072	0.858
6	7	0.783	2323.118	14175.450	0.132	0.380
7	7	1.000	2665.696	28239.914	-0.014	0.662

Figure 6.1: The inferred CPMM for the front place of articulation.



Transition		Probability of Transition	Parameter # 1		Parameter # 2	
from state	to state		Mean (M)	Variance (σ)	Mean (M)	Variance (σ)
1	3	0.221	777.974	7244.006	8.576	0.577
1	2	0.779	783.254	7430.651	8.587	0.574
2	4	0.354	392.941	507329.094	-0.655	0.447
2	3	0.326	289.525	482435.750	-0.693	0.460
2	2	0.320	183.919	495948.813	-0.752	0.469
3	8	0.000	962.596	878899.875	-0.591	0.431
3	6	0.095	632.267	569753.625	-0.576	0.416
3	5	0.146	393.876	470285.313	-0.629	0.453
3	4	0.728	429.573	518125.344	-0.637	0.443
3	3	0.031	308.035	499673.813	-0.681	0.462
4	8	0.000	1705.290	547970.375	-0.573	0.445
4	6	0.358	922.124	426738.938	-0.405	0.342
4	5	0.532	681.043	330763.125	-0.388	0.357
4	4	0.111	760.141	401895.469	-0.402	0.358
5	7	0.572	1014.282	419429.719	-0.399	0.338
5	6	0.363	960.179	421785.219	-0.398	0.342
5	5	0.065	725.972	329779.094	-0.379	0.351
6	8	0.150	1901.754	441204.188	-0.575	0.455
6	7	0.818	1033.238	419284.938	-0.398	0.337
6	6	0.033	1000.254	415977.719	-0.393	0.338
7	8	0.828	2081.247	274714.938	-0.617	0.481
7	7	0.172	1222.221	396869.563	-0.390	0.362

Figure 6.2: The inferred CPMM for the central place of articulation.



Transition		Probability of Transition	Parameter # 1		Parameter # 2	
from state	to state		Mean (M)	Variance (σ)	Mean (M)	Variance (σ)
1	2	0.550	570.090	1851.513	8.809	0.191
1	3	0.450	525.140	2384.136	8.438	0.369
2	3	0.459	206.060	29406.381	-0.350	0.020
2	4	0.541	253.393	3334.415	-0.257	0.140
3	4	0.878	393.276	19150.404	-0.168	0.134
3	5	0.122	1883.019	4183.322	-1.531	0.024
4	4	0.323	880.184	67645.172	-1.941	0.833
4	5	0.576	1819.642	153428.703	-2.176	0.517
4	6	0.101	2701.823	15439.756	-1.966	0.061
5	5	0.443	2218.530	96298.328	-2.202	0.708
5	6	0.557	2737.890	8447.664	-1.614	0.809

Figure 6.3: The inferred CPMM for the back place of articulation.

Table 6.1: Results on the recognition of the place of articulation

	F-B alg.	F-B alg.+rules	
BP	95%	97%	
CP	98%	98%	training set
FP	98%	99%	
BP	84%	94%	
CP	98%	98%	test set
FP	96%	96%	

to the decision process.

6.2 - EXPERIMENTS ON THE RECOGNITION OF DIPHTHONGS

Diphthong recognition was executed using the operators defined in section 4.1, 4.2, and 4.4.

6.2.1 - TEMPORAL RELATIONS OF FREQUENCY DESCRIPTORS

Frequency descriptions based on the alphabet $\sum_{PA}$ (see section 4.1.3) can be generated at fixed or variable length intervals. For each interval, the three places of articulation (FP, CP, BP) can be hypothesized and a score can be attached to each descriptor. The score is the a-priori probability computed for that descriptor by the Markov Models (see section 4.2). Segmenting the speech pattern into variable length intervals can also be done by putting an interval bound whenever a consistent change is detected in any of the pattern lines.

Figure 6.4 shows the spectral lines of a pronunciation of the letter U (ju) spoken in isolation. The highest scored hypothesis about the place of articulation is indicated for every 10 ms fixed-length interval. Symbols represent places of articulation for vowels according to an alphabet $\sum_{V}$ defined later on.

The redundant descriptors shown in Figure 6.4 suggest two possible approaches to ASR. By labeling fixed-duration intervals with the PA symbol having the highest score and considering this label together with others generated

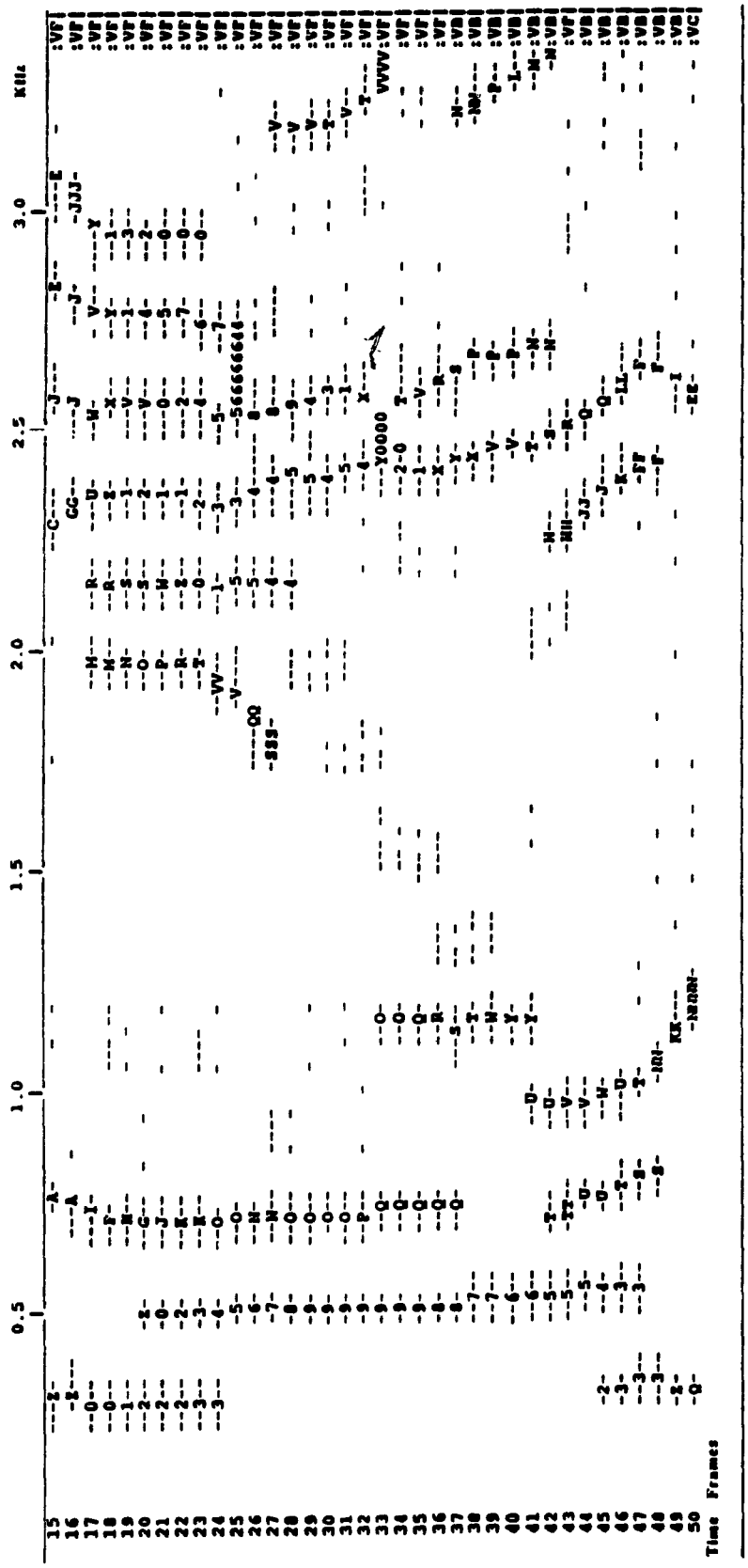


Figure 6.4: Spectral lines of a pronunciation of the letter U (ju).

by similar procedures but representing different phonetic properties, a **composite description** can be obtained for each interval and used as the output of a **vector quantizer.** This approach is suitable to be used with stochastic decoding [8] which is essentially **model-driven.**

Another possibility consists in having a **data-driven** segmentation followed by actions that generate scored hypotheses for each segment. This possibility has been investigated to produce the results of section 6.1.

The scored hypotheses of the segments in Figure 6.4 can be combined in order to obtain word hypotheses. If the lexicon is made of letters and digits, most of the lexical hypotheses contain vowels that can be identified only on the basis of the place of articulation. Furthermore, if letters and digits are pronounced in isolation, most of the sonorant portions of energy peaks correspond to vowels and diphthong. For this purpose, the alphabet used for descriptions in Figure 6.4 is defined as follows:

$\sum_V$: { (VB:Back Vowel), (VC:Central Vowel), (VF:Front Vowel) }.

The letter U can be characterized in terms of the places of articulation of its vowels as follows:

U= (VF) (VB)

and its sccre can be obtained by just multiplying the scores of VF and the scores of VB.

If the segment has three intervals, then the scores for the following possibilities have to be considered:

$$\{ (VF)(VF)(VB) \} , \{ (VF)(VB)(VB) \}$$

and the maximum of the scores has to be ascribed to the word hypothesis U.

It can be easily seen that the number of possibilities to be considered grows with the length of the segment.

The best probability for the hypothesis {(VF)(VB)} can be computed by using a Markov Model in the time domain like the one in Figure 6.5. The probability to be estimated is:

$$P_U = P(spectral\text{–}lines/((VF)(VB)))$$

An estimation of P_U is obtained by applying a Viterbi-like algorithm to the source of Figure 6.5 with the assumption that the probabilities of symbols associated with arcs are not known a-priori but are computed using CPMM in the frequency domain. Notice that the model of Figure 6.5 contains a rudimentary constraint on durations. In fact, q_{23} represents the probability that the duration of the VF segment is 20 ms, while q_{22} represents the probability that the duration of the VF segment is longer.

Table 6.2, shows the sequences of places of articulation for vowels for which time-domain Markov Models have been derived. The digit-letter vocabulary is defined in Table 3.1.

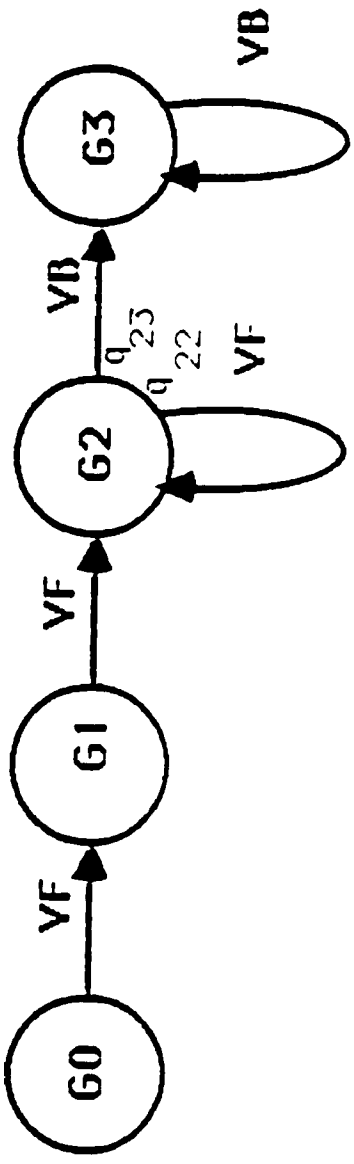


Figure 6.5: Time-domain Model for U (VF VB).

Table 6.2: Sequences of places of articulation and corresponding words

Sequence of places of articulation	confusion set
VF	E,G,P,3,V,C,B,T,D (h) L,M,N,S,F (t) A,K,J (h,d) X,H,8 (t) 6 (h,t) Z,7 (h,t)
VC	R
VB	2,O,4 (h)
VF VB	0,Q,U (h)
VC VF	I,5,9 (h,t)
VB VC	1
VB VC VF	Y
VC VF VB	W

It appears from Table 6.2 that the recognition of the place of articulation for vowels and diphthongs is not sufficient for an unambiguous characterization of all the words of the letter-digit vocabulary.

Confusion sets are indicated in the second column of Table 6.2. Each line of words corresponds to a confusion set. Confusion sets corresponding to the same sequence of PAs are identified and solved in many cases by analyzing the head(h) or the tail(t) of the word pattern and by generating hypotheses about consonants. These hypotheses are also scored by a-priori probabilities as described in [18], [22], [61]. Each conflict set in Table 6.2 ends with the indication of the head or tail analysis that will allow disambiguation among its components.

Details of the head and tail processes capable of performing such disambiguations are described in [18], [22], [61].

A more reliable disambiguation could also be performed with a recognition of the **manner of articulation** as well (for example: o of 4 vs u: of 2).

Not all the pronunciations of diphthongs are perfect; this may cause more ambiguity than expected by just considering theoretical places of articulation for vowels and diphthongs.

Let us consider now the letter A (aei). It is a diphthong, but the place of articulation of its phonemes is VF for both. So, even if a pronunciation of A has been segmented into many intervals, the hypothesis for A is the one corresponding to VF in all the intervals. The hypothesis will receive the same score as the hypothesis for E which also corresponds to VF in all intervals. In

order to solve this ambiguity, as well as the ambiguities generated by imperfect pronunciations, more detailed temporal relations among spectral lines have to be considered. These relations are introduced at the third level of description hierarchy and are used by a disambiguation process indicated by d in Table 6.2.

Level-3 descriptions refer to **temporal relations** of level-1 descriptions representing lines that are close in frequency. They are of the following type:

$$b_m : R_m(y_{m1}, y_{m2})$$

where R_m is a relation symbol, y_{m1} and y_{m2} are line descriptions.

R_m symbols belong to an alphabet $\sum_3$ whose elements are defined using stylized pictures in Figure 6.6.

Level-3 descriptions are obtained by an algorithm, ALDESLEVEL-3, whose details are omitted for the sake of brevity.

Composite descriptions can be obtained between Level-3 descriptions and PAs.

The following experiment shows how Level-2 descriptions can be useful for recognition. The purpose of the experiment is that of showing the potential of the property "follow-down" (FDN) for characterizing the diphthong /aei/.

The results shown in Table 6.3 refer to the pronunciations of E and K in letters like (E,P,B,T,D,K, etc.) for five speakers (four pronunciations for each letter and each speaker).

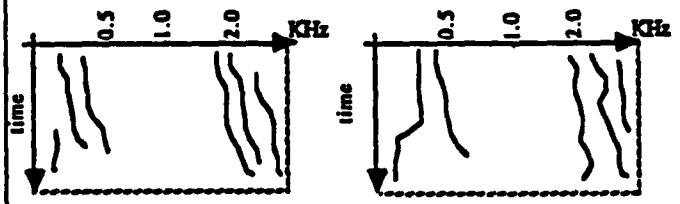
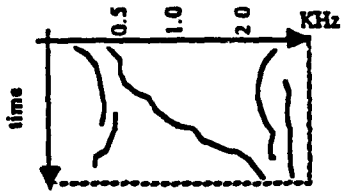
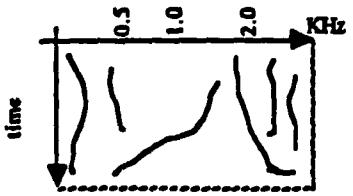
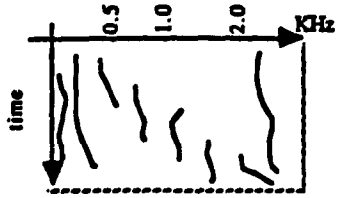
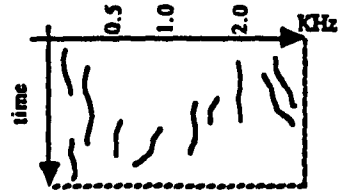
Components	comments	stylized patterns of properties
FDN	anchor line shifts downward along frequency	
ASND	lines grow upward along frequency from low-to-mid or from mid-to-high frequency bands	
DSND	lines grow downward along frequency from mid-to-low or from high-to-mid frequency bands	
STCR	Ascending lines are not continuous and appear as stair cases	
STCL	Descending lines are not continuous and appear as stair cases	

Figure 6.6: Definition of spectral morphologies

Table 6.3: Distinction of diphthong /aei/ as opposed to vowel /i:/

Speakers	Letter	FDN	VF	ΔF1/ae/	ΔF1/i/
#1	/k/	100%	100%	425-450	325-345
	others	0	100%	—	345-525
#2	/k/	100%	100%	475-525	350-375
	others	0	100%	—	425-525
#3	/k/	100%	100%	375-450	300-325
	others	0	100%	—	300-500
#4	/k/	100%	100%	575-650	325-350
	others	0	100%	—	355-610
#5	/k/	100%	100%	475-525	275-300
	others	0	100%	—	325-550

The strong evidence of follow-down property and 100% presence of VF in letter /k/ allows to distinguish the diphthong /aei/ and to unambiguously distinguish the first confusion set for VF in Table 6.2 from the second one. Table 6.1 shows also the frequency intervals in which spectral lines of /ae/ and /i/ involved in the VF relation were detected. As these intervals overlap, it appears doubtful that context-free recognition algorithm can be efficient in multispeaker detection of the diphthong /aei/ as opposed to the vowel /i/.

The results obtained show that hierarchical descriptions are powerful tools for detecting and recognizing diphthongs as opposed to single vowels.

Unfortunately, when many speakers are analyzed, even robust properties may disappear. In order to make the decision more reliable, a **redundant set of Transient Properties** (TP) are extracted based on acoustic preconditions. When these properties are detected, transitions of spectral lines are extracted covering a time interval whose duration can vary. For each line transition, five parameters are considered, namely $f_s - P_{11s}$ (the frequency of the starting point in the base-line), Δf (the frequency excursion of the transition), T (the time excursion of the transition), Δa (the amplitude excursion of the transition), $e_m - P_{12m}$ (the difference between the average energy in the line transition and the average energy of the corresponding segment of the base-line).

CPMMs have been derived for these types of transition using the five above mentioned parameters. The a-priori probability rendered by each transition CPMM was then multiplied by the probability obtained for the places of

articulation using the Viterbi-like algorithm in the time domain. In this way, a composite score for each sequence of vocalic places of articulation shown in the first column of Table 6.2 was computed. This composite score takes into account models derived for quasi-stationary patterns as well as models of transients. A complete score for each word hypothesis is then obtained by multiplying the composite vocalic score by the scores computed on the head and tail of the segment. The letter scores may refer to generic vocalic hypotheses or detailed consonantal hypotheses.

6.2.2 - RESULTS ON VOWEL AND DIPHTHONG RECOGNITION

The data described in section 6.1 were used for learning CPMMs for the place of articulation of vowels.

Recognition tests were performed on 20 new speakers who pronounced the vocabulary of isolated letters and digits described in section 6.4.3 and shown in Table 3.1. A decision was made by selecting the sequence of places of articulation for which the composite score made of probabilities of stationary and transient components was maximum.

The confusion matrix obtained with the above described experiments is shown in Table 6.4. Results are in percent. Some errors, like in the case of (VB) and (VF)(VB), are corrected by the recognition of consonants at the lexical level.

Figure 6.7: Typical error of the recognition method

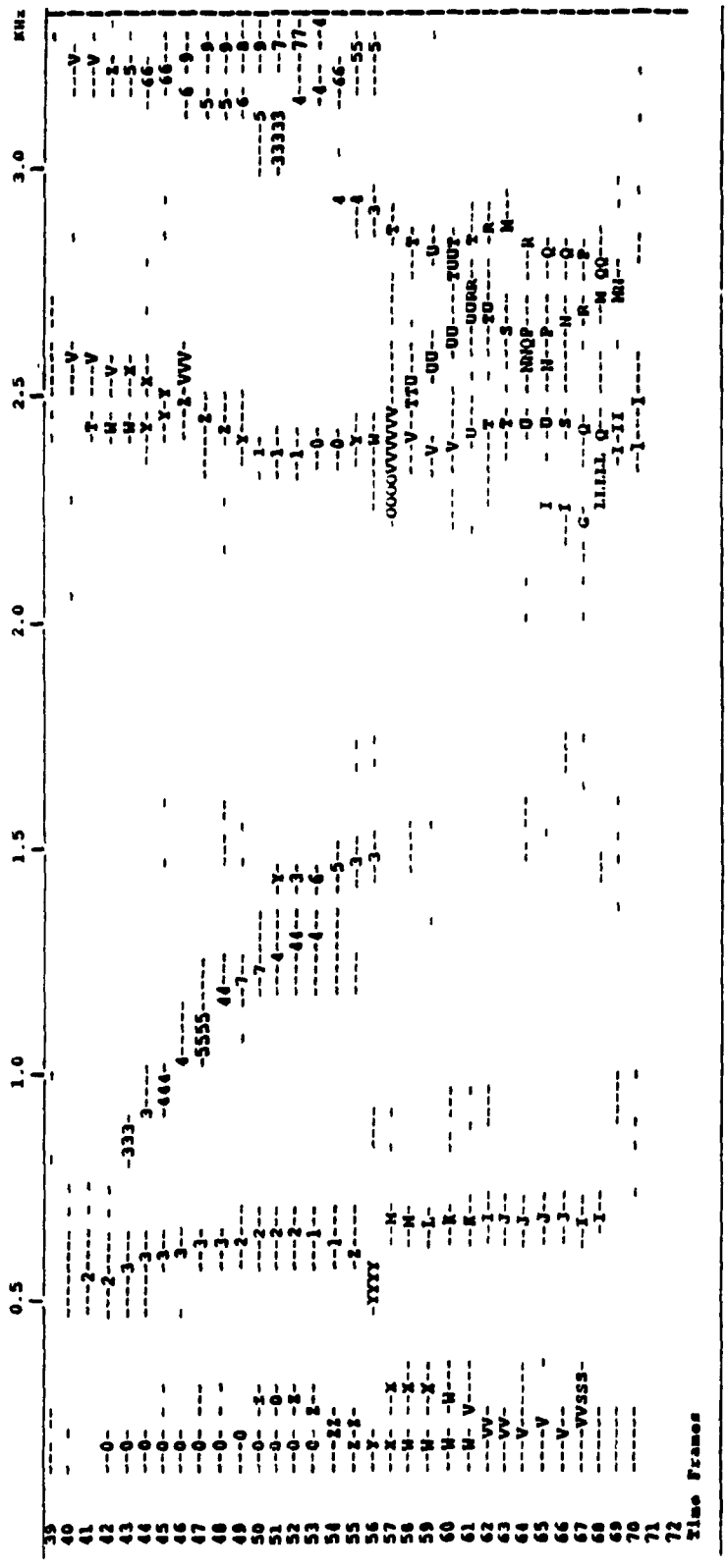


Figure 6.8: Typical error of the recognition method

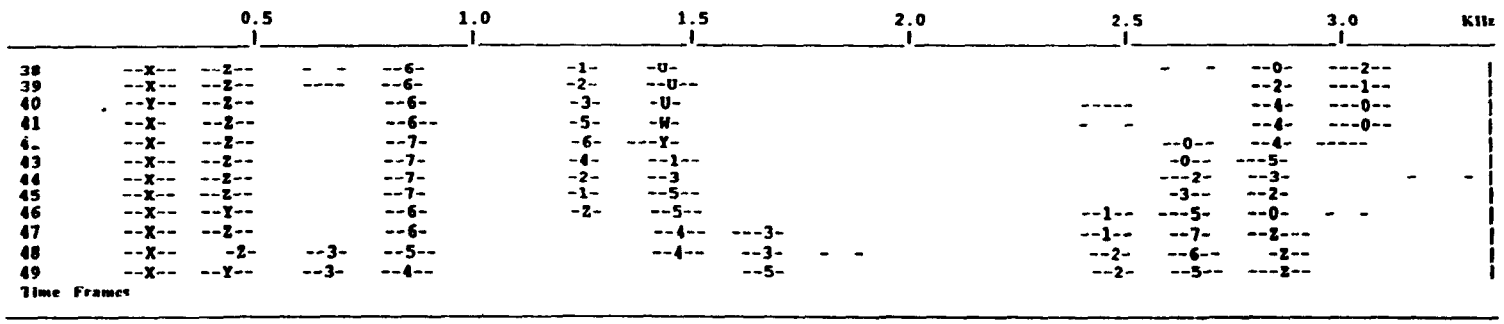


Figure 6.9: A quasi-stationary speech pattern and its sequence of vectors

Bi1	Bi2
891	9.257
-621	-1.595
-430	-1.279
405	-0.646
611	-1.122
1639	-0.858
1824	-0.519
2027	-0.742
2241	-0.959

Table 6.4: Confusion matrix for the recognition of vowels and diphthongs.

PRON / REC	VF	VC	VB	VF VB	VC VF	VB VC	VB VC VF	VC VF VB
VF	97	15	12	22				
VC		85			5			
VB	1		88	14				
VF VB				64				32
VC VF	2				95	20		
VB VC						80	6	
VB VC VF							94	
VC VF VB								68

The most typical errors are shown in Figure 6.7 and Figure 6.8.

Figure 6.7 shows a pattern of Y (/uai/) for which the final /i/ is missed.

Figure 6.8 shows a pattern of 1 /uan/ for which the initial /u/ is missed.

The results show the effectiveness of the use of spectral lines and performance models of their distortions in the recognition of sequences of places of vowels.

6.3.1 - EXPERIMENTS ON THE RECOGNITION OF VOWELS AND DIPHTHONGS

We will concentrate, in the rest of this section, on the recognition of vowels and PAs in quasi-stationary, non-nasalized speech intervals. The operators that have been used to produce these results are explained in section 4.1, 4.2, and 4.4.

6.3.1 - PARAMETER CHARACTERIZATION

Speech segments corresponding to vowels extracted from the pronunciation of letters, digits and words containing 10 English vowels have been used. In order to learn statistical relations of SLs, a learning set was prepared in which vowel labels were assigned to segments using an automatic procedure made possible by the choice of words used for learning.

Figure 6.9 shows a speech interval with the corresponding vector Y as defined by (4.6).

Details on learning and recognition of vowels and PAs will be given in the next sub-section.

6.3.2 - EXPERIMENTS ON THE RECOGNITION OF STATIONARY SEGMENTS

In order to investigate the possibility of using spectral lines and CDHMM for ASR an experiment has been set up for the recognition of English vowels. A signal data-base has been built by asking 20 speakers (10 male and 10 female) to pronounce the monosyllabic words shown in Table 6.5. Each speaker read a randomly ordered list which included 40 occurrences of each word from Table 6.5. Every pronunciation of every word was then processed using a network of HP workstations including one HP 9000-236 especially equipped for speech processing, an HP 9000-320 and an HP 9000-330. Table 6.5 contains also a 5 word vocabulary containing vowels that are common to number of Languages other than English.

Task decomposition among units was performed as suggested in [20]. Fourier Transformation, Primary Acoustic Cues as defined in [20] and Spectral Lines were computed for each word in roughly 10 times real-time.

For each word pronunciation, three vowel samples were automatically extracted using PAC description. A vowel sample was extracted in the middle of the vowel in an interval of 60 ms duration.

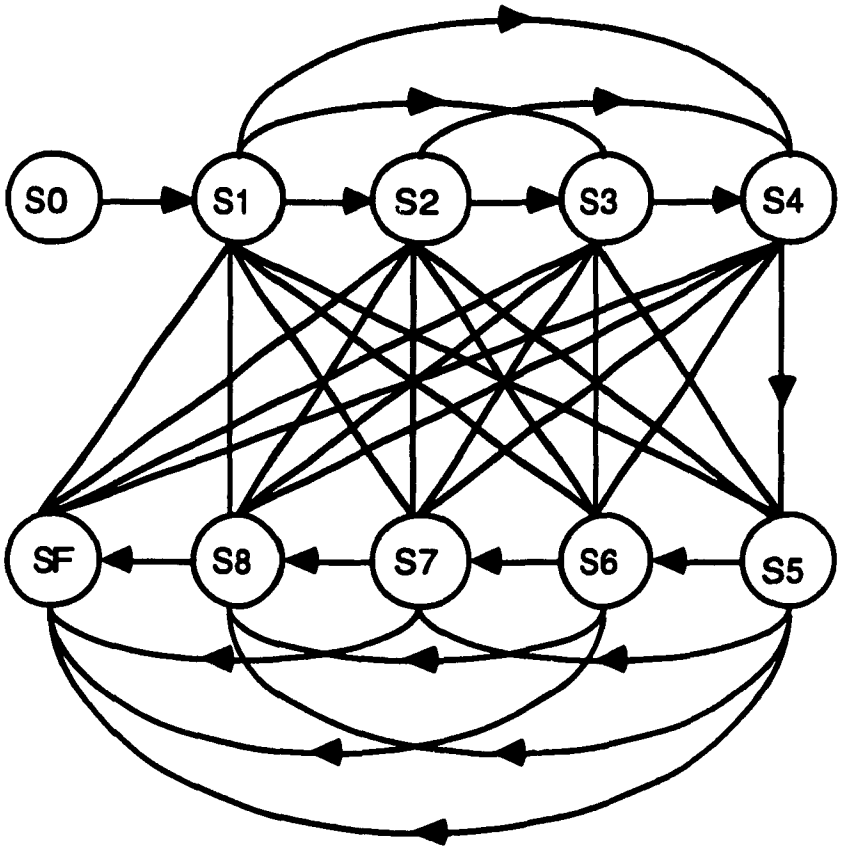


Figure 6.10: General structures of CDHMMs for the recognition of vowels.

Table 6.5

VOCABULARY FOR VOWEL RECOGNITION	
5 VOWELS	**10 VOWELS**
bed	bat
beep	bed
boot	beep
but	boot
saw	but
	far
	fur
	pit
	put
	saw

Table 6.6: Transition probabilities of a CDHMM

startpoint	endpoint	probability	m_1	v_1	m_2	v_2
0	1	1.0	318.1	675	6.57	0.081
1	2	0.01	117.0	855	0.313	0.006
1	4	0.01	1445.0	130502	-1.669	0.174
1	5	0.406	1696.8	12652	-0.924	0.923
1	6	0.573	1978.5	1781	0.567	0.138
2	6	1.0	1992.0	248004	1.072	0.072
4	6	1.0	1953.0	238388	0.7	0.031
5	6	0.821	2048.1	8774	0.539	0.173
5	7	0.179	2533.4	3228.5	0.325	0.057
6	7	0.966	2592.9	4604	0.811	0.121
6	8	0.034	2890.3	48880.5	0.383	0.663
7	8	0.286	3082.9	6570	0.106	0.409
7	9	0.714	3275.0	18647	0.319	0.118
8	9	1.0	3490.2	14750	0.023	0.220

where:

m_1 = mean of frequency

v_1 = variance of frequency

m_2 = mean of amplitude

v_2 = variance of amplitude

Learning was performed using data from 10 speakers (5 male and 5 female). Recognition was performed using data from the other 10 speakers.

Markov chains were built in the following way. The frequency range from .1 to 3.5 kHz was subdivided into intervals. A basic chain was built by considering a linear sequence of a state and a transition, each transition corresponding to a frequency interval. Other transitions were then added in order to allow each state to reach any of the states following it.

Figure 6.10 shows the general structure of a CDHMM for the recognition of vowels as it is set up before starting a learning phase.

A transition t from state s_n to state s_k that produces vector y_{ij} is associated with the mean of the difference between the frequency and energy of the n-th line and the frequency and energy of the base line. Each transition is also associated with a transition probability not shown in Figure 6.10 for the sake of simplicity.

At the beginning all the transitions having the same destination state are associated with the same means and standard deviations.

Chains conceived with the above mentioned criteria have been constructed and used for learning and testing vowel models. A tabular description of a Markov chain for a front vowel is given in Table 6.6.

The first experiment concerned learning and recognition of the place of articulation as defined by (4.4) for the five vowel vocabulary of Table 6.5.

Table 6.7

recognition results for vowels

mode / task	multispeaker CDHMM	sp. independent CDHMM	sp. independent MLN
place of articulation	97.1	95.2	96.9
5 vowels	97	95	96.6
10 vowels	73.6	69.9	87

A second experiment concerned learning and recognition of the five vowel vocabulary.

Other two experiments were performed using half of the data from each speaker for learning and the other half for recognition. The task of the latter experiments was learning and recognition of the place of articulation and of the five vowels in a multispeaker mode while the task of the first two experiments involved the same classes in a speaker-independent mode. Finally, two other experiments have been conducted in the speaker-independent and the multispeaker mode using the 10 vowel vocabulary.

The results of these experiments are summarized in Table 6.7. The results in Table 6.7 clearly show that spectral lines and CDHMMs are more than adequate for the recognition of the place of articulation and for vowels having remarkably different place or manner of articulation. This suggested the use of such an approach for the recognition of vocabularies for which discrimination of vowels having close place and manner of articulation is not required. Such an application will be further discussed in the following Sections.

Nevertheless, the recognition of 10 vowels was not performed satisfactorily with the above proposed method. In order to try to improve the recognition performances of the 10 vowels attempts were made to introduce discrimination rules for cases characterized by relations like in (4.18). In order to avoid the tedious work of manually inferring rules by experiments, another learning and recognition paradigm was tried based on Multi Layered Networks (MLNs). The

reason for such a choice is that MLNs allow one to perform competitive learning and to discover pattern regularities.

These aspects were found particularly attractive for the case of vowels because some of them are so similar that competitive learning is a more suited paradigm for discovering regularities that enhance differences among pattern classes.

Furthermore, MLNs can perform speaker normalization by learning functions of SLs in accordance with hypotheses made by other researchers that speaker normalization should involve relations between formant frequencies.

A detailed definition of MLNs and a discussion on their use in ASR can be found in [5]. Only a brief introduction of the MLN used for the recognition of vowels will be given in the following.

Figure 6.11 shows the general scheme of an MLN. The input layer is fed by a **Property Extractor (PE)**, that acts as a window analyzing the data with variable time and frequency resolution. PEs may also extract data from the speech waveform. The MLN in Figure 6.11 has two hidden layers and one output layer.

The PEs are mostly rectangular windows subdivided into cells as shown in Figure 6.11.

In our case, PE is a column of 64 windows of 60 ms duration. Each window has a frequency width of 50 Hz.

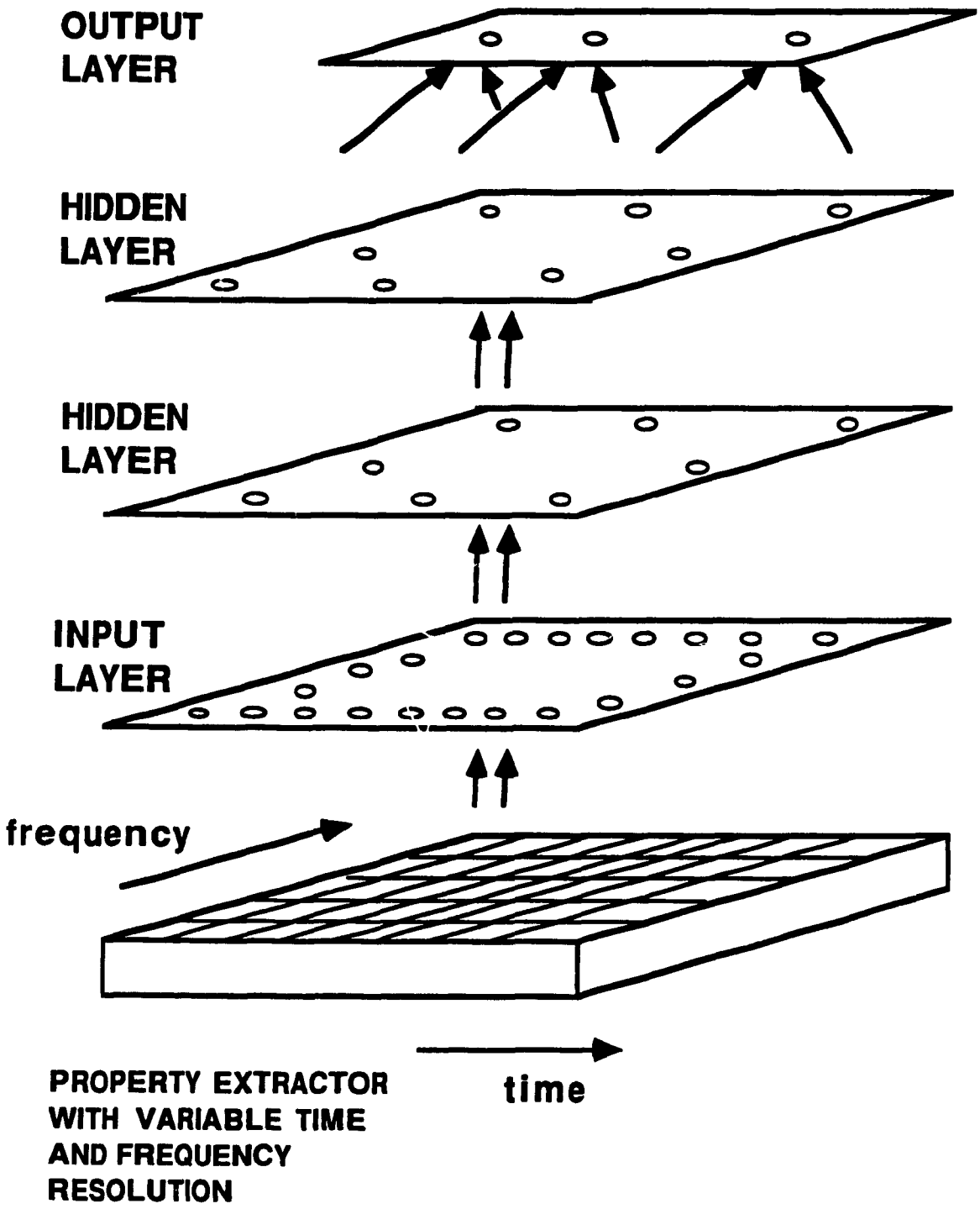


Figure 6.11: Structure of a Multi Layered Network (MLN)

In a first experiment 64 spectral samples where sent to an MLN with 40 nodes in the first hidden layer, 20 nodes in the second hidden layer and 10 nodes in the output layer.

The weights of the connections among nodes were learned using the Error Back Propagation Algorithm (see [77] for details).

An error rate of 20.4% was obtained. A second experiment was executed by using only spectral lines coded as proposed in [5] using an MLN with 320 input nodes and 200 nodes in each hidden layer. An error rate of 18.8% was obtained showing that SL are a good coding of speech spectrograms. The two MLN outputs where combined together using heuristic rules inferred from the training set and an error rate of 13% was obtained.

The obvious conclusions are that SLs contain enough information for discriminating among vowels and MLNs are powerful tools for performing speaker-normalization.

Furthermore, competitive learning shows remarkable advantages especially when the task requires fine discrimination.

Further efforts are required in order to evaluate the possibility of using MLNs for more complex tasks. For this reason the task described in 6.2 was implemented using stochastic models in the time domain.

6.4 - EXPERIMENTS ON THE RECOGNITION OF LETTERS AND DIGITS

The application of the EPN model to the recognition of letters and digits has been detailed in section 5.1. The experimental results are presented in the next sections.

6.4.1 - EXPERIMENTAL ENVIRONMENT

Let's assume we want to characterize sequences of letters and digits according to the lexicon defined in Table 3.1 with a little pause between them. Let us also assume we want to represent knowledge that is speaker-independent.

The *EPN* conceived for this purpose is based on a **data-driven** approach (action execution is decided based on data) and has several levels.

The highest level is represented in Figure 6.12. It consists of a "push" arc to a subnetwork LEX representing the lexicon (Figure 6.13). The arc following state $S\phi 1$ in Figure 6.12 is associated with a PUSH action to a subnetworks that returns the probability of having a PAUSE after the just analyzed AS. From state $S\phi 2$ an iterative jump to state $S\phi$ under the condition "not-end" represents the fact that there is still a part of the input signal to be analyzed; otherwise the recognition process will stop on the execution of a POBABS arc with a stop function associated to it.

It is subdivided into three subnetworks depending on the number of ASs detected per word. A subnetwork ASH for generating SU hypotheses for each

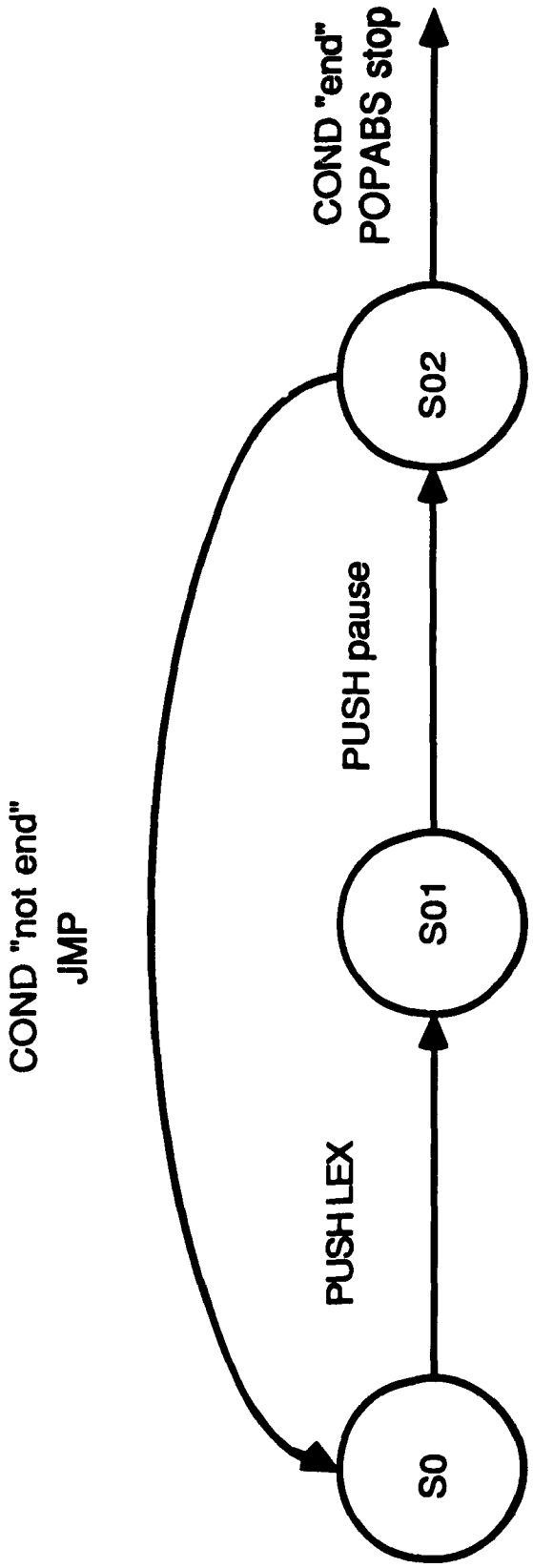


Figure 6.12: Top level of an EPN for the recognition of a lexicon

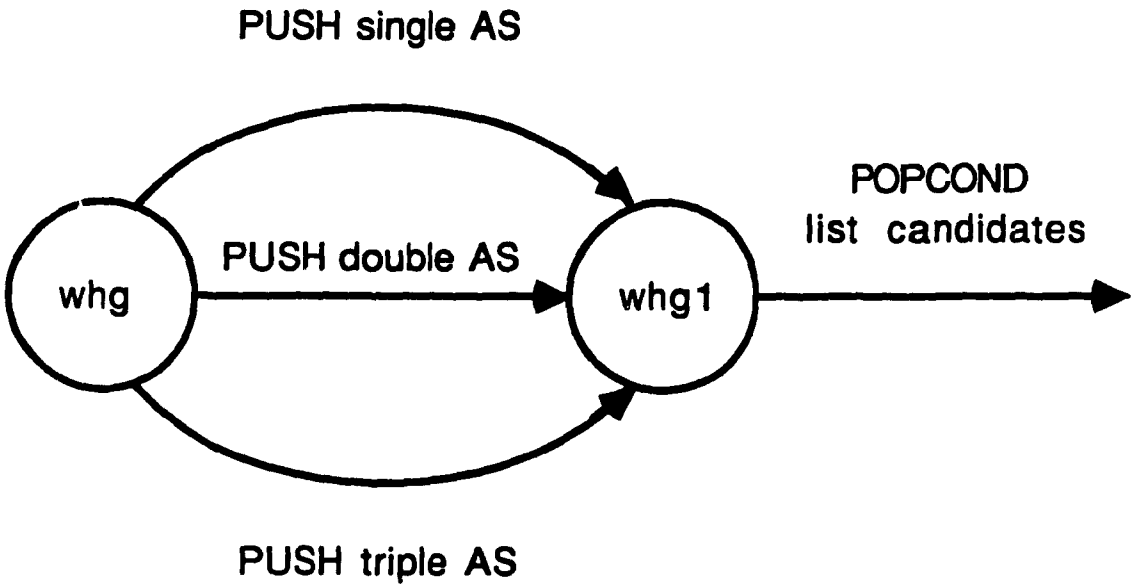


Figure 6.13: Subnetwork LEX

Figure 6.14: Subnetwork for AS hypotheses

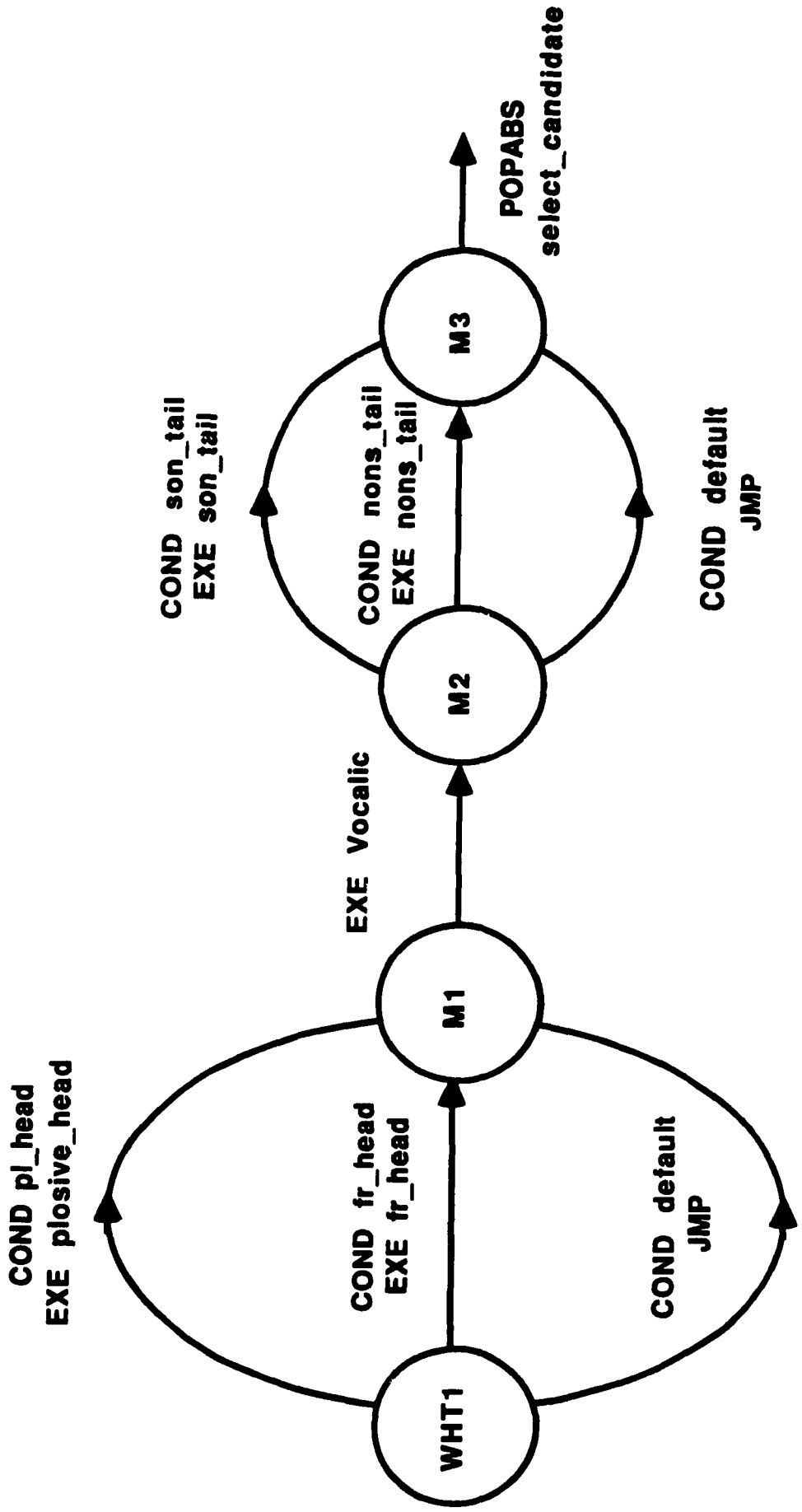


Table 6.8: Definition of Primary Acoustic Cues (PAC)

Primary Acoustic Cues

Symbol	Attributes	Description
LPK	tb,te,ml,zx,	long peak of total energy (TE)
SPK	"	short peak of TE
MPK	"	peak of TE of medium duration
LOWP	"	low energy peak of TE
LNS	tb,te,zx	long nonsonorant tract
MNS	"	medium nonsonorant tract
LVI	tb,te,ml,zx	long vocalic tract adjacent to a LNS or a MNS in a TE peak
MVI	tb,te,ml,zx	medium vocalic tract adjacent to a LNS or a MNS in a TE peak
LDD	emin,tb,te,zx	short deep dip of total energy
LMD	"	long dip of total energy with medium depth
SMD	"	short dip of total energy with medium depth
LHD	"	long non-deep dip of total energy
SHD	"	short non-deep dip of total energy

Attribute description

Attribute	Description
tb	time of beginning
te	time of end
ml	maximum signal energy in the peak
emin	minimum total energy in a dip
zx	maximum zero-crossing density of the signal derivative in the tract

AS is shown in Figure 6.14. Two subnetworks similar to the one in Figure 6.14 are available for words with 2 ASs. Three subnetworks are used for words with three ASs.

ASH subnetwork uses actions for analyzing the "head", the "vocalic part" and the "tail" of each AS. The AS head is analyzed by **attached procedures** (actions) performing an **Elaboration-Decision (ED)** paradigm. Let us call these types of procedures **ED-actions**. ED-actions perform **variable-depth analysis** on intervals of AS and will be described in the section "The Elaboration - Decision paradigm". There are two possible ED-actions for the head of an AS, namely:

-plosive head

-fricative (including affricate) head

The choice of the ED action is made by disjoint conditions associated with arcs. These conditions are regular expressions of Primary Acoustic Cues (PAC) introduced in [20]. PACs are descriptions of the time evolution of speech loudness combined with the description of speech intervals where energy is predominant in frequencies greater than those corresponding to the most prominent sonorant resonances. A definition of PACs is recalled in Table 6.8.

A **data-driven** process DDP1 performs signal acquisition, Fast-Fourier Transformation (FFT), signal description in terms of PACs and segmentation of the speech signal into ASs. For each AS a process is created that instantiates ASH.

For the AS for which an ASH instantiation has been created, the signal, its FFT and the PAC description are stored into the WM associated to the ASH instantiation.

Condition "pl-head" is a regular expression of PACs corresponding to acoustic morphologies that are cues for plosive sounds.

The condition "fr-head" is analogous but it contains morphologies that are cues for fricative sounds. If none of the two above mentioned conditions is verified, then the **default** condition is satisfied. State M1 in Figure 6.14. will be reached by only one arc. As it will be seen later, the ED action for the AS head identifies the AS head as a possible acoustic subsegment and extracts some acoustic properties. Using techniques partially described elsewhere [21], hypotheses about the place and manner of articulation for the speech unit in the head subsegment are generated and scored by the following a-priori probability:

$$Prh = Prdata(h)/place \cap manner \tag{6.1}$$

"place" is a variable that takes values in the following set:

$$PLset = \{labial, front, central, back\} \tag{6.2}$$

"manner" is a variable that take values in the following set:

$$MNset = \{high-vowel,\ low-vowel,\ oral,\ nasal,\ unvoiced,\ voiced-nonsonorant\} \tag{6.3}$$

The description data(h) contains acoustic properties the system knowledge considers worth to be extracted given the suprasegmental characteristics of the head subsegment. These properties can be broad-band spectral energies for a

fricative head or transient descriptors for a plosive head.

In the case of letters and digits, the place and manner of articulation are not identified separately. Speech Units hypothesized by head actions are phoneme symbols, when they are important for recognizing the letter or the digit. For example, only the head actions can distinguish between P, B, T, G and all the letters beginning by a consonant and ending with the same vowel. In this case the consonants are SU hypothesized by the head process. For the letters or the digits beginning with a vowel, it is not the head action that has the proper knowledge for distinguishing among vowels, thus a generic "vocalic" SU is hypothesized for all these words.

For each head action, suitable acoustic properties are extracted and a-priori probabilities are collected in a learning phase for concatenations of acoustic properties corresponding to each SU symbol that can be hypothesized as an AS head.

For example, "plosive head" executes a network of actions described in [21]. These actions extract a number of acoustic properties and produce hypotheses not only about plosive sounds but also about any SU (including "vocalic") that may have generated the observed properties. After state M1, the ED-action "vocalic" is executed. It segments the vocalic part of AS into stationary and transient units.

Let

$$v_1 v_2 \ldots v_x \ldots v_X$$

be such subsegments. For each segment v_x spectral lines are considered as data (see [62] for details) and a-priori probabilities about place and manner of articulation are obtained by HMMs of spectral lines in the segment v_x. For each subsegment and for each consistent "place-manner" pair, the following probability is computed.

$$Pr(v_x)=Pr(data(v_x)/place \cap manner) \tag{6.4}$$

From state M2 to state M3, ED-actions for the tail of AS are executed similar to those used for the head. A probability

$$Prt=Pr(data(t)/place \cap manner) \tag{6.5}$$

scores the hypotheses of the tail subsegment. The data extracted in the head, the subsegments of the vocalic part and the tail can be assumed to be independent. The "select" action associated with the POPABS arc computes the probability for each candidate hypothesis

$$\text{Pr (data /hyp).}$$

where "hyp" indicates the sequence

$$P_{h1} P_{h2}, \ldots, P_{hi}, \ldots, P_{hI}$$

of the SU hypotheses and selects the best candidate.

Let us consider the hypothesis /b/. The hypothesis "hyp" can be represented by the following sequence of two SUs:

$$\text{/b/ = (B)(E)}$$

where /b/ represents the letter, B and E represent SUs. The corresponding probability will be computed as follows

$$Pr(data/b)=Prh_B\left[\prod_{x=1}^{X} Pr_E(v_x)\right]Prt_{NT} \qquad (6.6)$$

where Prh_B is the left-hand side of the (6.1) computed for the place and manner of articulation of B, Pr_E is the left-hand side of the (6.4) computed for the place and manner of articulation of E and Prt_{NT} is the probability that the segment has no tail. Probability Prt_{NT} is set to 1 if the default action is taken in the transition from M2 to M3.

Duration statistics for the Speech Units involved in each hypothesis can be collected and used in the "select" action. As probabilities for places and manners of articulation are computed in well delimited time intervals, durations of these intervals can be considered as additional data.

The segmenter that produces ASs may undersegment and, very rarely, oversegment. In both cases, hypotheses are considered with one, many or no vocalic segments. A maximum of two and a minimum of zero segments are allowed both for head and tail.

Figure 6.15 shows an example of the pronunciation of "five". The total energy curve as well as the time evolution of the low-to-high frequency energy ratio are shown with the corresponding PAC description. Vertical lines delimit the head, vocalic and tail subsegments.

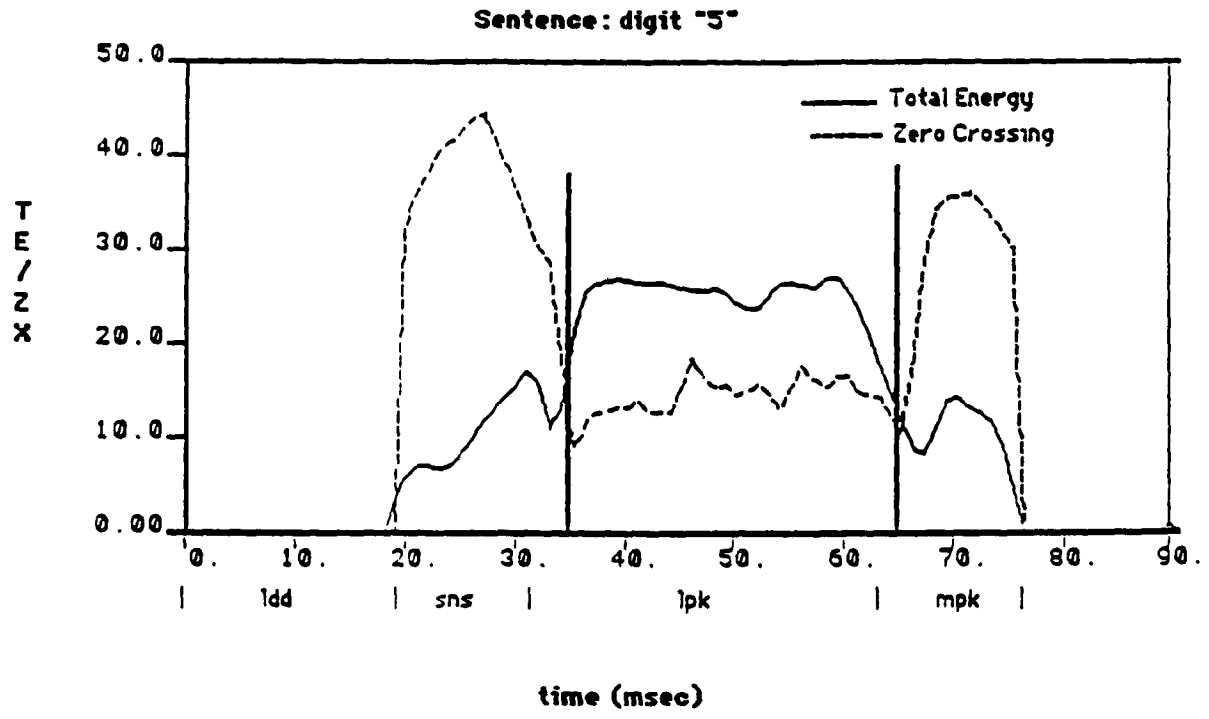


Figure 6.15: Head-vocalic-tail segmentation for a pronunciation of "five".

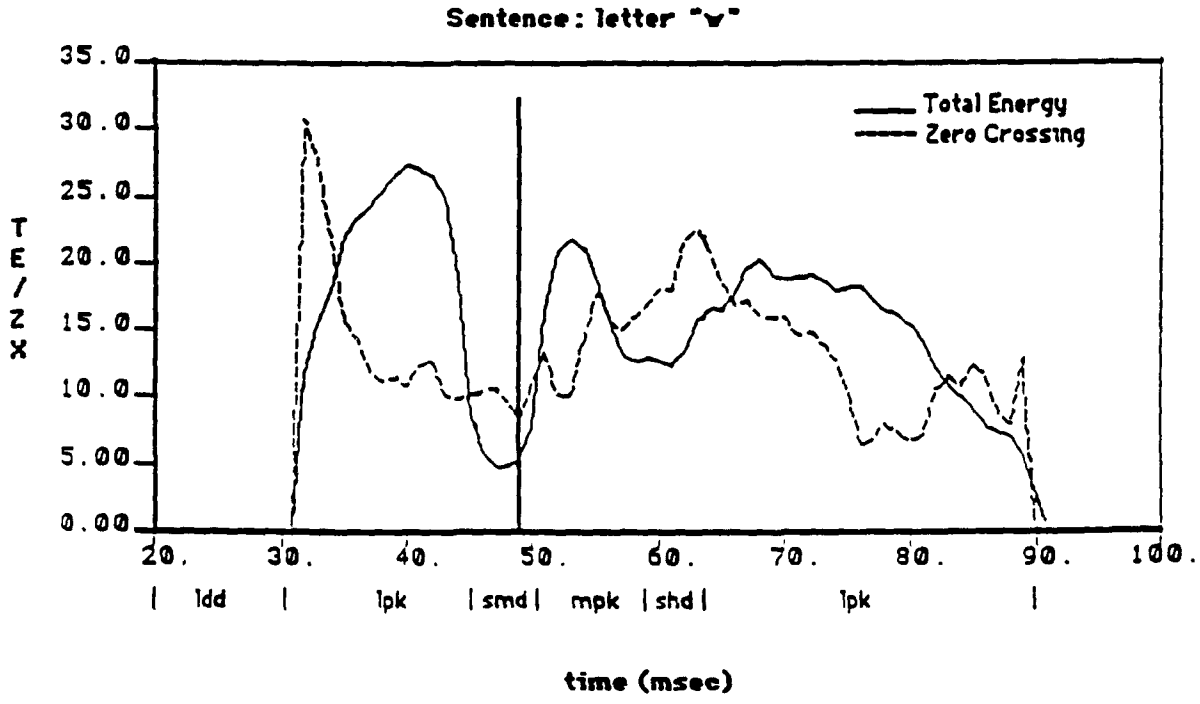


Figure 6.16: Example of a two-segment pronunciation

The first candidates produced by the "fr-head" action are:

f 0.0078
t 0.0003
.
.
.

The first candidates produced by the "vocalic" action are:

AI 0.269
UAI 0.198
EH 0.161
.
.
.

The tail has been treated as a sonorant one but hypotheses for all the sounds observed under such conditions are generated.

The first candidates produced by the tail action are:

v 0.001
n 0.0002
.
.
.

The action POPABS f has selected the first candidate of the following list:

5
4
Y

Figure 6.16 shows an example of pronunciation of /w/ split into two ASs delimited by a thick vertical line.

More details on the ED-actions are reported in section 6.4.2.

6.4.2 - THE ELABORATION-DECISION PARADIGM

An Elaboration-Decision (ED) paradigm is executed through a certain number of ED cycles which analyze descriptions of the **same** acoustic segment at different levels of depth. The use of the ED paradigm has been discussed in a previous paper [21]. ED cycles are PNs. According to this approach $d_t\ (\tau_t)$ in (3.6) is made of **components** extracted at different levels of depth. Thus, $d_t\ (\tau_t)$ can be represented as follows:

$$d_i(\tau_t)=d_{i\phi}\ (\tau_t)\ d_{i1}\ (\tau_i)\ldots d_{iJ}(\tau_i)\ldots d_{iJ}(\tau_t) \tag{6.7}$$

J being a function of i,*i.e.*$J=J(i)$.

Description $d_{i\phi}\ (\tau_t)$ is made of Primary Acoustic Cues (PAC). These cues are extracted by a spontaneous activity.

The first ED cycle executes an Elaboration Phase (EP) that computes a first level description $d_{i1}(\tau_t)$. A Decision Phase (DP) is executed that uses as data the following description:

$$d_i(\tau_i)=d_{i\phi}(\tau_i)d_{i1}(\tau_t) \tag{6.8}$$

DP decides the next action based on the available descriptions. Other ED cycles are performed until a termination condition is reached. Scored hypotheses are associated with descriptions.

Local scores are a-priori probabilities. For example, the score of hypothesis H in τ_i is:

$$Pr(d_i(\tau_i)/H)=Pr(d_{i\phi}(\tau_i)/H)Pr(d_{i1}(\tau_i)/d_{i\phi}(\tau_i),H)) \quad (6.9)$$

Different ED-actions may extract different data and obtain different descriptions for the same time interval τ_i.

There are possible solutions to this problem. The one which has been adopted for letters and digits recognition consists in using homogeneous descriptions (PAC sequences) for selecting the best path of actions. Each one of these paths applies the ED-paradigm only once for each signal subsegment. Thus the probabilities computed for each candidate hypothesis in a subsegment are based on the same data for all the considered hypotheses.

Another problem arises from the fact that local probabilities may not be "homogeneous" in different intervals τ_i because they have been computed on descriptions extracted at different levels of depth. There are different possibilities for combining local probabilities. One of them consists in "summarizing" the states of each local conflict of hypotheses by a symbol belonging to a summary Alphabet $\sum_{SA}$ and using strings of summaries for building a **performance model.** Such a performance model can be a Markov Source. A model for each word can be constructed and used together with a Language Model for computing sentence likelihoods like in [8]. The other possibility, that has been adapted for the letters-and-digits protocol, consists in just multiplying the probabilities of segments in a sequence to obtain a cumulative score.

A detailed description of the properties used in the ED paradigm for recognizing letters and digits appears in other papers describing motivations for

choosing certain properties and the experimental conditions in which statistics have been collected [21], [68], [75]. A brief summary of these properties is given in the following.

6.4.2.1 - PLOSIVE SOUNDS

Various properties extracted both in the time and frequency domain have been used. They are described in [21].

The first description level consists of PAC. The second description level contains buzz-bar and burst indicators extracted from the time waveform, its envelope, the time evolution on certain frequency bands.

The third description level contains burst properties extracted in the time intervals in which burst indicators were detected.

The fourth description level is related to spectral line transitions at the voice onset.

6.4.2.2 - VOCALIC INTERVALS

Spectral lines are extracted with an algorithm described in [62]. Markov Models are used for modeling statistics of frequency and energy of spectral lines. They generate probabilities of place of articulation for intervals of 20 ms duration. Sequences of hypotheses are kept in a search space in which a node represents a sequence like:

$$(v_1{:}vf,\ v_2{:}vf,\ v_3{:}vb,\ v_4{:}vb)$$

where f stays for "front", b for "back" and v for "vowel". The node sequence corresponds to the hypothesis that subsegments v_1 to v_4 represent a sequence of a front vowel followed by a back vowel. The product of the probabilities of each subsegment hypothesis is used as score for the node.

6.4.2.3 - OTHER CONSONANTS

Liquid and nasal sounds are hypothesized using a mel-scaled filter bank and considering time evolutions of energy differences in a continuous parameter Markov Model. Other levels of descriptions involve the use of Markov Models for spectral lines in the stationary zone of the consonant and in the transient segments in order to capture statistics of properties discussed in [19]. More details on the content of this subsection can be found in [75].

The ED-paradigm corresponding to the subnetwork "nons-tail" PUSHed in the transition between state M2 and state M3 in Figure 6.14 will be described as an example. The condition "nons-tail" for the protocol of letters and digits is the following regular expression of PACs:

$$COND\text{''}nons-tail\text{''}=\alpha(LPK+LVI)(\beta+D)(NS+SP)\delta \qquad (6.10)$$

where α is any noise description that can precede the PACs of a vowel, β is any short non-deep dip, D is any deep dip, NS is any descriptor of frication noise, and δ is any noise description at the end of a word.

Equation (6.10) corresponds to a "Level-ϕ" description obtained by the first elaboration phase. The first decision phase selects a subnetwork NT1 if the actual description matches with β and NS (/f/ or /s/ are expected), a subnetwork NT2 if the actual description matches D (/x/, /h/ or /8/ are expected).

Let us describe NT1 in detail. The elaboration phase of NT1 selects the speech signal in the time interval corresponding to the description that matched NS. In this time interval a power spectrum in the 2-8 kHz band is obtained every 10 ms using a filter bank with 8 filters. Filter bandwidths grow logarithmically from low to high frequencies.

From the filter outputs, five parameters are computed every 20 ms:

1) G: the center of gravity of the power spectrum,

2) I: the filter index for which the energy output, is maximum,

3) the first three cepstral coefficients defined as follows:

$$c_s = \sum_{i=1}^{8} \log_{10} \left[\frac{E_i}{\sum_{l=1}^{8} E_l} \right]^{b_i} \cos\left(s \frac{2i-1}{8} \frac{\pi}{2} \right) \qquad (6.11)$$

where: s=(1,2,3), E_i is the energy at the output of the i-th filter, b_i is the filter bandwidth.

The decision phase uses three Continuous Parameter Markov Models (CPMM), one for each hypotheses NT1 is supposed to generate, namely, /s/, /f/ and /ʃ/. The 3 CPMMs have three states each. Various transitions between states

are allowed. Each transition is associated with the mean and the variance of the five above mentioned parameters which are assumed to be independent and with Gaussian distribution.

Learning has been performed on 15 speakers using the Forward-Backward algorithm [4]. The same algorithm is used for generating scored hypotheses.

The conjecture that such parameters are good properties for characterizing fricative sounds is based on Speech Science and previously acquired experimental evidence. The example described so far refers to a variable depth analysis with two levels of depth.

6.4.3 - EXPERIMENTAL RESULTS ON LETTERS AND DIGITS

A corpus consisting of 400 pronunciations of the vocabulary defined in Table 3.1 was used for evaluating the ASR model built based on the theory presented in this paper. The corpus was obtained by asking 100 (50 male and 50 female) speakers to pronounce four times the entire vocabulary.

Speakers were mostly university students and instructors with different mother tongue. They were all asked to speak in English.

A computer program generated random sequences of 5 letters or digits. Each speaker was asked to pronounce each sequence presented to him/her with a little pause between each letter or digit. Data were acquired with a Hewlett Packard Special Purpose Workstation HP 9000-236.

Signals were sampled at 20kHz over 12 bits. The signal was windowed by a 23 ms Hamming window, and a 128 points Fast Fourier Transformation (FFT) was computed every 10 ms by a TMS 320.

The rest of the processing was carried out on a VAX 8600, while a distributed version of the recognition system following a paradigm proposed in [20] is under development. This new version uses a TMS 320 and the two processors of the HP 9000-236 and HP 9000-320 Workstations.

Learning was done on the first 40 speakers for properties extracted by head and tail actions. Markov sources inferred with previous data for the experiment described in [62] were used for recognizing vowels and diphthongs.

Twenty other speakers (10 males, 10 females) were used for testing. A sample for each word of the 36-word vocabulary was used for each speaker resulting in 20 samples for each vocabulary word.

Experimental results are summarized in Table 6.9.

Column headings represent pronounced words, row headings represent recognized words.

Figure 6.17 shows an example of an error. The signal envelope of the pronunciation of a/b/ is plotted versus time. As there is no buzz-bar preceding the vowel onset, the letter has been recognized as /e/.

The remaining data were used for performing other experiments on groups of letters, like those belonging to the E-set (E,G,P,3,V,C,B,T,D), the A-set

(A,K,J,8,H) the I-set (I,9,Y) the AE-set (L,M,N,F,S) and others both in a multispeaker mode (40 speakers, different utterances) or in a speaker-independent mode. Remarkably better performances were observed for the multispeaker with respect to the speaker-independent mode (more the 10% higher recognition scores for some groups).

This suggests that the generalization power of the methods proposed in this thesis applied to our acoustic properties has performances that can be improved if the number of speakers used for learning is greater than 40. We suspect that differences between the multispeaker and the speaker-independent mode will be noticed even if the number of speakers used for learning is greater than 100.

Table 6.10 shows the results obtained by different researchers on similar speech recognition problems. The table is organized in such a way that the vocabulary, the number of speakers, the speaker independence, the strategy, and the recognition rate are shown. The complexity of the vocabulary increases from eleven digits to vowels, letters, and letters and digits. The power of the recognition strategy increases from Markov models to simple networks, trees, and AI environment with specialized matchers (EPN with operators). Simple strategies are inadequate to model the speaker independent recognition process of a complex vocabulary such letters and digits or and that is the reason why their use is limited to digits or letters only.

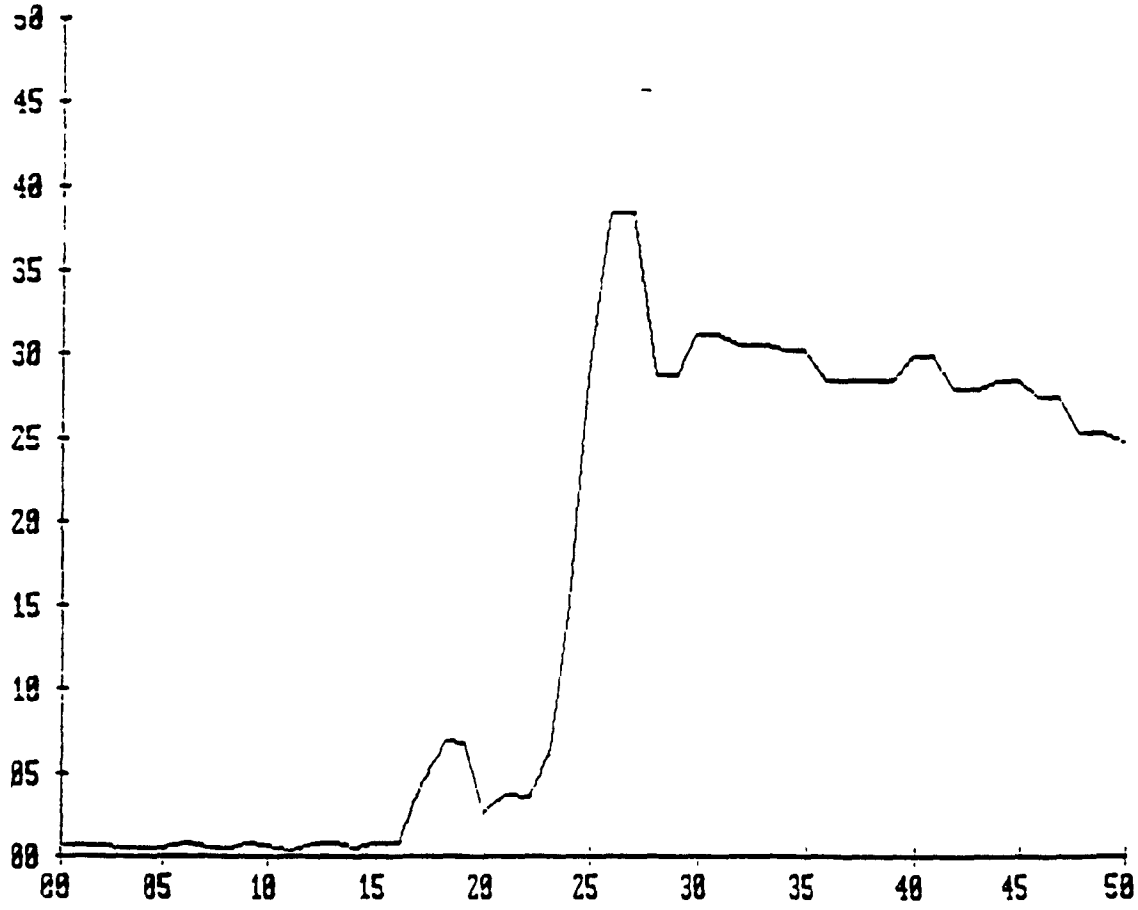
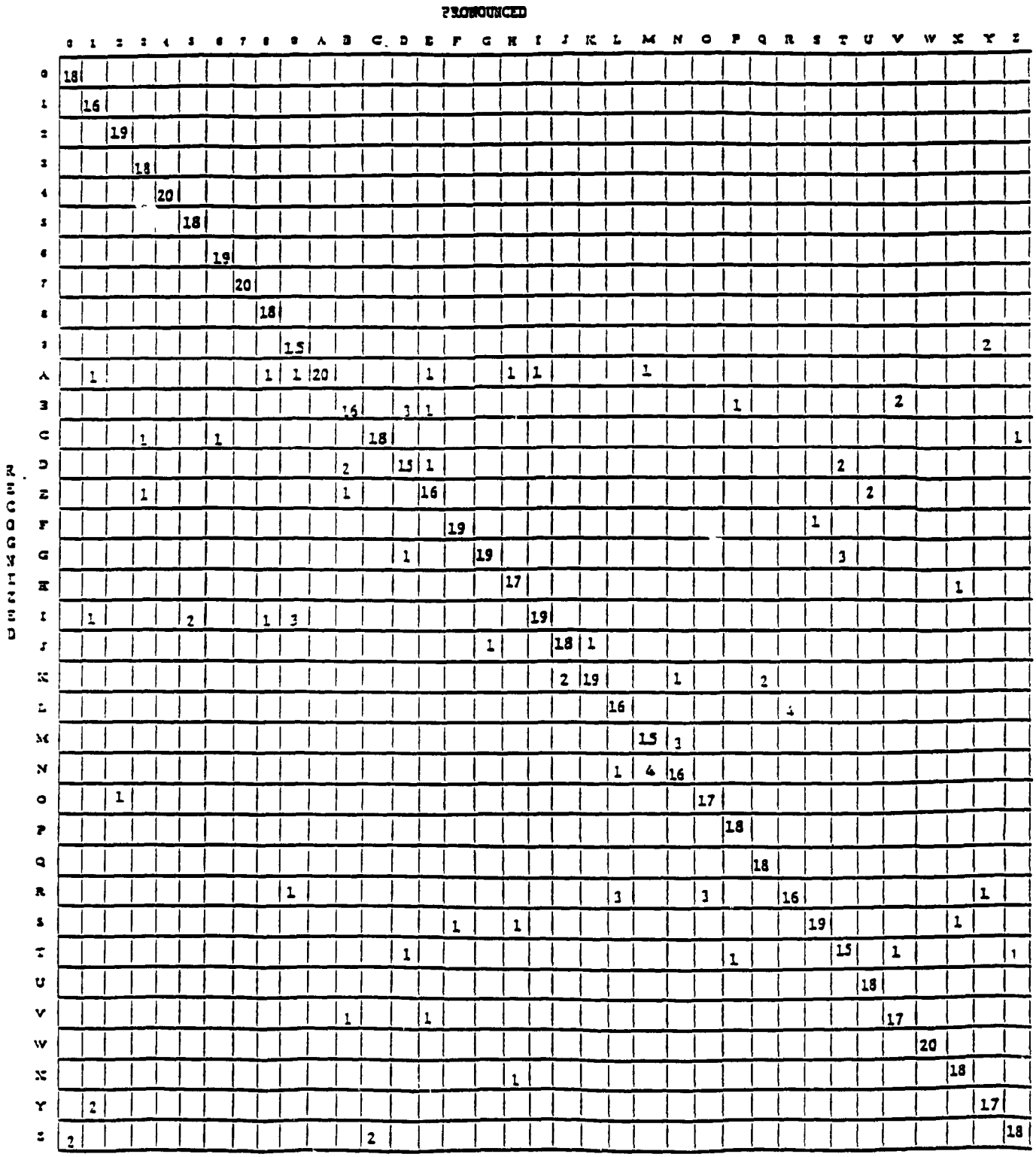


Figure 6.17: Time evolution of signal envelope of the pronunciation of /b/

Table 6.9: Experimental results for letters and digits recognition



Authors	Kopec Bush 85	Cole Stern 83	Stern 87	McGill May 87	Rabiner Dec 87	McGill 88
vocabulary	eleven isolated digits	isolated letters	isolated letters	isolated letters + ten digits	isolated letters + ten digits	vowels
# speakers	225 (TI2)	20 10m 10f	20	60	100	40
speaker independence	yes	yes 19 + 1	speaker adaptation (adjust statistics)	yes 40 + 20	multi speaker	yes 20 + 20
strategy	simple Hierarchical network + matchers	decision tree + feature extractors	improved 83	A.I. environ. specialized matchers	Markov models	Markov models + neural networks
recognition rate	97-99	89.5	93.4	88.2	88.5	87-96.6

Table 6.10: Comparison with other research works

6.5 - LEXICAL ACCESS PERFORMANCE AND PERSPECTIVES

Preliminary results, obtained running experiments on lexical access whose strategy has been presented in section 5.2, show a performance similar to the one obtained in [57] using the same data and Markov models. A much better performance can be achieved in the author's opinion by exploiting the facilities on the EPN to represent a more detailed knowledge for lexical access. Two major points are worth to be investigated.

1) Dictionary description and organization

2) Phonological rules

1) The description of the dictionary in terms of only PPF is quite rough. Tools are available [68], [75] to recognize phonetic units with a finer grain of detail. Vowels can be identified [23], and so can fricatives, plosives etc. To make use of this knowledge and tools, the dictionary should report this description. The EPN could then handle a lexical access to a finer discrimination. Figure 6.18 shows the detailed description of the word "most" and Figure 6.19 shows the correspondent EPN. Variable-depth analysis is then used to stop the recognition process whenever the hierarchical level is fine enough to avoid any ambiguity.

2) The description of a word is not unique. The same word can be pronounced in different ways due to geographical variations or change in the population or statistical distortions. The dictionary should include the descriptions corresponding to the possible variation of a word. Then the EPN could infer thorough its learning algorithms, the probability of these variation and its

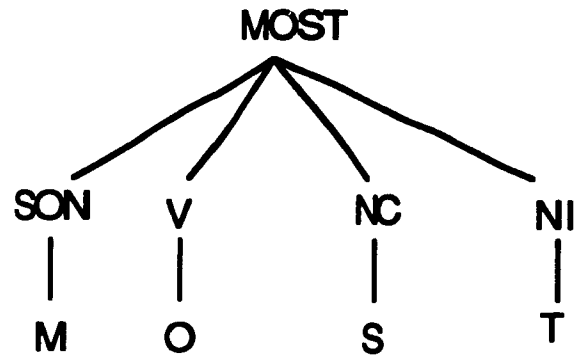


Figure 6.18: Hierarchical description of the word "MOST"

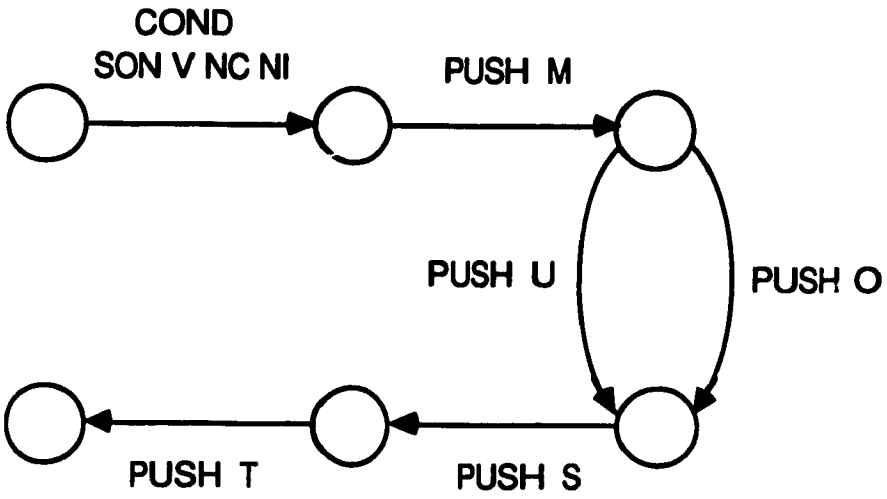


Figure 6.19: Subnetwork for the lowest level description of the word "MOST"

existence in a certain population. Automatically a lexical access which is learnt for a particular geographical area or population (example the Eastern North America) can be produced.

6.6 - EXPERIMENTS ON THE RECOGNITION OF ISOLATED DIGITS

Different speaker produce speech according to different speaking modes. A feature-based word model taking into account all the possible inter-speaker variations would be very complex. The EPN model, together with the neural net operator (see section 4.5), allows the fine grain description of words to express the inter-speaker variations and coarticulation distortions into a rich model with low computational complexity. Conditions are put on the arcs in such a way that paths that are not likely to have generated the signal are not considered at all. This is possible because the feature based speech analysis is highly related to the articulation parameter, and these are subject to physical laws ruling their behaviour in time - they cannot vary too sharply, for example. Therefore fast executed conditions on symbols and durations actually prune the model almost to a unifilar one. The pruned model can then be very detailed and can perform sophisticated analysis because of its reduced complexity. Conditions at this level can be regarded as heuristic functions that lead the supervisor toward the solution disregarding unlikely paths. Of course optimality is lost for higher efficiency. The approach has been tested on isolated digits belonging to the vocabulary of letters and digits (refer to sections 6.4 and 6.4.3) and the 96% overall recognition rate

has been achieved on 20 speakers. The model corresponding to the digit "9" is shown in Figure 6.20. The transitions of such a model are labeled with a regular expression as described in [14], and with a time interval "m - n". For example the regular expression for the transition from state 2 to state 4 is "(f+F)L" and the time interval is from 14 to 16. Time interval means that there can be no less than 14 frames activating the transition and more than 16 frames have exponentially decreasing probability. Time intervals are expanded according to the model in Figure 4.7.

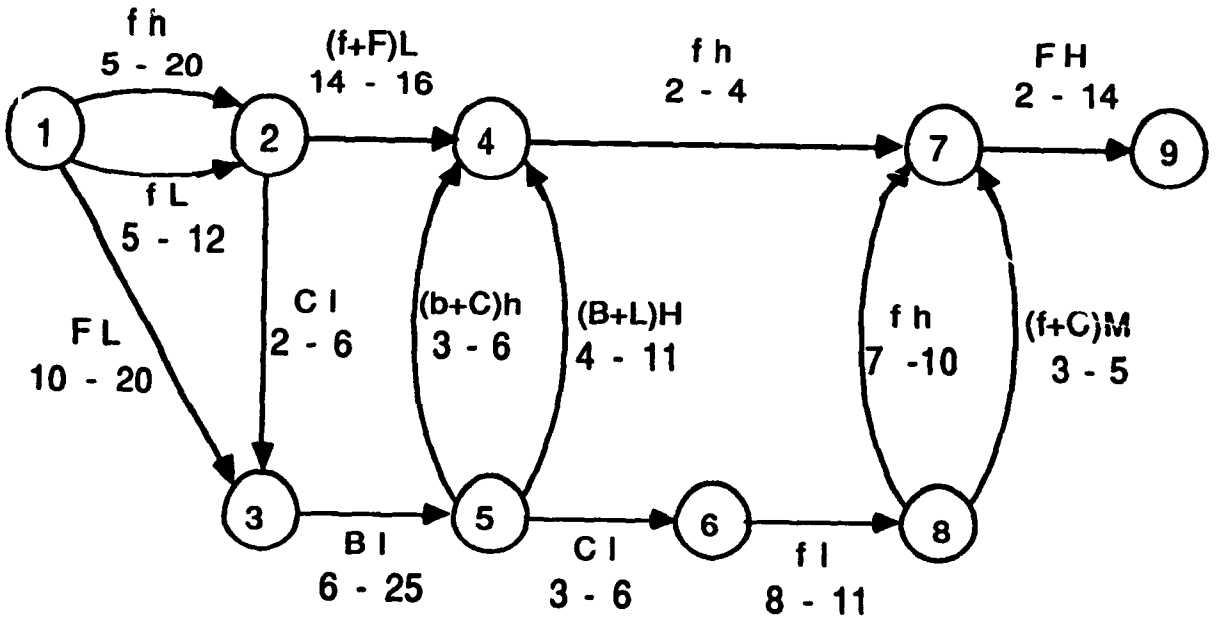


Figure 6.20: Digit model

CHAPTER - 7

7 - CONCLUSIONS

7.1 - CONTRIBUTION TO KNOWLEDGE

The EPN model constitutes an original and new contribution. The novelty is in the flexibility of the EPN to represent and implement intelligent strategies together with strongly numerical algorithm from pattern recognition. The EPN fills the gap between systems that were based on numerical strategies and had little or no intelligence, and systems that were inherently intelligent, but lacking of sophisticated numerical support. Furthermore many AI strategies are simply a special case of the EPN model. In this sense the EPN can do what other systems do and much more. Frequently a system is bound to a certain degree of sophistication that depends on the technology. The EPN model offers a really open ended and extensible system, very flexible for research purposes in which the solution algorithm has to be found, and were the strategy is not known a-priori, The EPN intrinsically encourages the definition of hybrid systems with the power of capturing the real knowledge to solve a problem, without the limits of a predefined paradigm.

The model has proven effective in modeling the strategy of recognition of different difficult problems in speech recognition. Compared to standard network approaches to speech recognition the EPN is way ahead from the point of view of underlying intelligent power [49], [51]. The results obtained in the applications show comparable or better recognition rates with plenty of architectural power to

investigate alternate strategies or solution.

The application of the EPN to the speaker-independent recognition of letters and digits proved effective. The strategy introduces a paradigm to merge into an intelligent model evidences coming from different sources. The research is still open for other solutions. The cognitive and feature-based approach showed its power because of the different focus of attention and variable-depth analysis required to distinguish among certain words where the difference appears in a very specific time interval.

Variable-depth analysis, the possibility of invoking subnetworks at different levels and of using Markov models by some subnetworks are among the novelties of the proposed approach with respect to previously proposed network-based models for ASR [49], [51].

The possibility of using acoustic properties as in [13] with stochastic models of their descriptions is another novelty of the proposed approach.

A first test of the proposed methodology has been performed on a difficult vocabulary spoken by a variety of speakers. Results are comparable to the ones recently obtained by other researchers [13] on a population of limited size and only on letters.

Results can be improved, especially for some letters, by collecting statistics of the acoustic properties of a larger speaker population or by adding new operators and observing their influence in the performances of the fully automated learning-recognition systems presented in this paper. This may indicate

how difficult is the proper characterization of different speaking styles when detailed phoneme discrimination is a requirement for word recognition.

Finally, a theory of machine perception of speech based on data-driven actions and a paradigm for its implementation based on extended procedural networks have been proposed.

The EPNs for lexical access fully support the needs to specify complex dictionary structure together with sophisticated recognition algorithms which integrate numerical or probabilistic techniques with rules and heuristics.

Neural nets training power combined with a Markovian time alignment seems promising for a higher performance and incremental recognition systems. If the desired recognition rate is not achieved, it is because the present features are not discrimating enough, hence the solution would be to add further features until the phonemes or the smallest recognition units are properly identified and classified.

EPNs define the strategy of application of neural nets operators. The EPN constitutes the AI environment in which the activation of neural net operators is triggered and the contributions of the operators are combined together in an intelligent way. The EPN can define an efficient non optimal stochastic process based on flexible and contextual heuristic functions. The limits of current stochastic systems [52] are the simplicity required by the word models to have a reasonable speed performance of the recognition process. Coarticulation and inter-cluster speaker variations would require a more sophisticated, finer grain description of the words. This does not seem to be practical in conventional

systems, but appears ordinary in a EPN environment. Knowledge and contextual heuristic functions are applied to the underlying stochastic process to reduce the space of the probabilities to a smaller non optimal subspace.

EPN and neural net operators provide a model of description and recognition that is as rich in detail as desired and computationally efficient.

The application of frequency domain-based Markov models on spectral lines is original and tries to capture into a stochastic model the inter-speaker variations for a speaker independent application. Traditionally, Markov models have been used for the time alignment of the speech signal. The novelty is in the idea of considering the frequency as the domain in which the sequence of input symbols is defined. Consequently new meaningful parameters have been defined in the frequency domain. The definition of the three classes back, central, and front for the classification of the place of articulation has shown the ability of achieving a high recognition rate and a granularity fine enough to make the distinction among diphthongs in the context of isolated speaker independent letters and digits recognition.

The results show the effectiveness of the use of spectral lines and performance models of their distortions in the recognition of sequences of places of vowels.

It is likely that a larger number of speakers would allow us to obtain a better characterization of spectral line distortions in quasi-stationary vocalic segments. Different **speaking modes** are likely to produce different distortions on expected

pattern morphologies.

As the system is rather robust, a systematic analysis of its error should suggest the use of other transition properties, a better implementation of the actions for extracting them and a better statistical characterization of their distortions.

To study the limits of this approach it has been tested on a larger number of classes. The experiments on the recognition of the English-American vowels, that were pronounced in a fixed context from many speakers, showed that the performance is still interesting in a multispeaker mode, but it degrades when speaker-independence is sought. Neural networks seem to be a better tool in such a context and for this kind of granularity (see section 6.3.2).

7.2 - IMPLEMENTATION

The EPN model and the software for the implementation of Markov model have been implemented in Pascal on a VAX-8650 running VMS. The EPN model required about 4000 lines of code and the Markov model routines required about 26000 lines. Additional 200 kbytes were used to store certified test cases. The software has been developed according to the rules reported in appendix A.1. The EPN model interfaces the user with two main routines: the parser that reads a file in which an EPN is described in the format shown in section 2.3 and produces an internal representation on the network, and a routine that executes a

particular network.

7.3 - FUTURE RESEARCH

The EPN model has been used for the application to speech recognition. Further researches can be made to exploit the paradigm of integrating different knowledge sources and various strategies into a hybrid paradigm. Indications for speech recognition are reported in section 5.3.1 and [6].

Other computer science research areas can benefit from the EPN model. Software Engineering, for example, is a very promising domain for the study and the application of Artificial Intelligence concepts. The EPN model can be applied to the modeling of the software process and other software engineering activities whose solution with monolithic strategies is still unsatisfactory. The flexibility and extensibility of the EPN model match the requirements of software engineering systems.

APPENDIX

A.1 - IMPLEMENTATION ISSUES

A Software Engineering Approach is particularly useful when the system to be developed and subsequently maintained reaches even a medium size and there are several people involved among users and designers. Although a Software Development Environment was not available a certain number of Software Engineering principles have been applied to the life cycle of the software involved in the research. The system has been design by top down development and refinements. Stub routines have been created to test the hierarchy. The system has been written mostly in standard Pascal as it is defined in the ANSI standard [1]. The standard has been verified (although not enforced) by the compiler itself. The non standard features include only lexical violation to the standards that can be arranged by some operation of batch editing (example, non standard use of "_" character) and few machine dependent features (non standard and non portable such as I/O and file access). This have been all enclosed in a file which have to be modified to transfer the software onto another machine.

A special testing procedure has been applied. Counters have been defined for all the branches of the software and a simple profiler has been written. Test cases and test output have been stored to be compared upon software modification in the maintenance phase. Guidelines based on a modification of the ones in the course [56] have been applied and are reported in Appendix A2.

A1.1 - CODING

- Adopt naming conventions for routines, functions, global constants and global types. Example:

  procedure x_yyy_nnnnnnn

  where:

  x = programmer identifier

  yyy = project/version code

  nnnnnnn = meaningful procedure name

- Meaningful names can make the code more readable. Remember: programs must be read by humans

- Coding should be simple. Tricky code is hard to read and error prone: keep it simple stupid

- Add the following prologue comments at the beginning of every procedure or function

```
(*****************************************************************
 *
   FUNCTION NAME:

   AUTHOR:                    DATE:

   DESCRIPTION:

   PARAMETERS:

   RETURN VALUE:

   GLOBAL AND EXTERNAL VARIABLES REFERRED:

   PROCEDURES AND FUNCTION CALLED:

   FILES ACCESSED:

   COMPILER:                          VERSION:

   HARDWARE/OS:
 *
 *****************************************************************)
(*****************************************************************
 *
   PROCEDURE NAME:

   AUTHOR:                    DATE:

   DESCRIPTION:

   PARAMETERS:

   GLOBAL AND EXTERNAL VARIABLES REFERRED:

   PROCEDURES AND FUNCTION CALLED:

   FILES ACCESSED:

   COMPILER:                VERSION:

   HARDWARE/OS:
 *
 *****************************************************************)
```

- Use, if necessary, a consistent abbreviation technique. For example, you may remove the vowels from the names

  m_mc7_truncate ---> m_mc7_trnct

- Alphabetize declaration lists

- Use parentheses to clarify expression and make the parse tree unambiguous (extra parentheses do not hurt and avoid errors)

  a * b * c / (d * e * f) ---> (a * b * c) / (d * e * f)

- Indent code

- Add identifiers in comments to explicitly match begin-end, then-else, case-end pairs:

```
while ... do
begin (* 1 *)
   if ...
   then (* 2 *)
      .
      .
      .
   else (* 2 *)
      case ...
         .
         .
         .
      end (* case *)
end (* 1 *);
```

- Select a data representation suitable for the problem (it is much easier to build a tree with records and pointers rather than arrays)

- Write modular code. Procedures should be about 60 lines or less since this fits on one or two screens and can be easily understood. A procedure should perform a single logical task

- AVOID USING GLOBAL COMMUNICATION IN MODULES SINCE THIS CAN HAVE RIPPLE EFFECTS THROUGHOUT THE ENTIRE PROGRAM WHEN A MODIFICATION IS MADE

- Use structured coding

- Use case statement rather than nested if's. Remember that standard pascal does not allow a default value when the case selector does not match any label. Make sure that illegal values do not get through the case statement

```
case i of:
   0:   ...;
   2,5:   ...;
   3:   ...
end;
```

becomes:

```
if i in {0,2,3,5}
then
   case i of:
      0:   ...;
      2,5:   ...;
      3:   ...
```

```
    end (* case *)
  else
    m_mc7_error(m_mc7_ill_case_select) (* exception handling
                    procedure *)
```

- Declare all variables (even if some languages allow default declarations)

- Initialize all variable before use (even if some systems and some compilers initialize the variables automatically). If this is not done unexpected values may be encountered when the program is run in another environment (debugger) or with another operating system.

- Use constants rather then numerical values

  for i := 0 to 10 do ... —> for i := m_mc7_min to m_mc7_max do ...

- Do not use the same variable name for different purposes

- Always assign a return value to output procedure parameters. This must be particularly taken into account in exception handling.

- Make use of exception handling if available, otherwise design your own exception handling mechanism. The latter would include error or recovery procedures together with consistent error coding and error propagation. A program is supposed to degrade gracefully upon exceptions and to stop only if no other actions can be taken

- Supply command files to regenerate the entire software system (compilation, library generation, linking etc.)

- Put all non standard and non portable calls (system calls etc.) in a module and redefine them in terms of user defined procedures. That module represents the only software to be rewritten in case of portability problems.

A1.2 - DEBUGGING

- Antibugging is better than debugging

- Use conditionally executed debugging statements or (better) conditionally compiled debugging statements.

CONDITIONAL EXECUTION:

```
type
   m_mc7_opt_exe = array[1..m_mc7_max_opt] of boolean;

var
   opt_arr: m_mc7_opt_exe;

      .
      .
      .

   if opt_arr[29]
   then
      writeln( ... );
```

CONDITIONAL COMPILATION:

```
begin (* 1 process_line *)
```

```
(*#-dbg       writeln('process line');*)
      if not eoln then
      begin (* 2 *)
            read(c);
(*#-dbg           writeln(c);*)
            if c = '(' then
```

the lines labeled with (*#-dbg .... *) are not compiled into the object code, but they are activated after preprocessing:

```
   begin (* 1 process_line *)
(*#+dbg*)   writeln('process line');
      if not eoln then
      begin (* 2 *)
            read(c);
(*#+dbg*)       writeln(c);
            if c = '(' then
```

- Suspect all data. Data consistency checks should be performed by every routine.

  Although such checking code slows the program it is better than producing incorrect results with a faster program

- Check array bounds

A1.3 - TESTING

- Test cases have to be designed in order to test:

  every instruction

  every branch

  every recursion and iteration

normal input

unusual input

extremes (acceptable input at the limit of its range)

exceptions (graceful degradation must be achieved)

- Compare output files with previous (and certified) output files

- Make a profile of your program (use conditionally executed or conditionally compiled counter increments if a profiler is not available)

- Constructed testing data is a good starting point, actual data adds reality

- Test your program again after any change (testing command files are helpful)

REFERENCES

[1] ANSI Standard Pascal Definition,
ANSI/IEEE 770x3.97 - 1983.

[2] Bahl L. R., Brown P. F., De Souza P. V. and Mercer R. L.,
Speech Recognition with Continuous-Parameter Hidden Markov Models,
Proc. ICASSP 1988, pp. 40-43, IEEE 1988.

[3] Baird H. S.,
Applications of Multidimensional Search to Structural Feature Identification,
in Proc. Nato Advanced Research Workshop on Syntactic and Structural Pattern Recognition,
Sitges, Spain, October 1986.

[4] Baum L. E.,
An Inequality and Associated Maximization Technique in the Statistical Estimation for Probabilistic Functions of Markov Processes.,
Inequalities, Vol. 3, pp. 1-8, 1972.

[5] Bengio Y. and De Mori R.,
Use of Neural Networks for the Recognition of the Place of Articulation,
Proc. IEEE Int'l Conference on Acoustic, Speech and Signal Processing,
New York, N.Y., pp. 103-106, 1988.

[6] Bengio Y., Cardin R., Cosi P., De Mori R., and Merlo E.,
Speech Coding with Multilayer Networks,
in: Neurocomputing - Algorithms, Architectures, and Applications,
Ed. by F. Fogelman, Springer-Verlag, to be published.

[7] Bengio Y., Cardin R., De Mori R., and Merlo E.,
Programmable Execution of Multi-Layered Networks for Automatic Speech Recognition,
Communications of the ACM, Feb 1989.

[8] Bhal L. R., Jelinek F., and Mercer R. L.,,
A Maximum Likelihood Approach to Continuous Speech Recognition,
IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-5, pp. 179-190, 1983.

[9] Biermann A. W. and Feldman J. A.,
On the Synthesis on Finite-state Machines from Samples of Their Behaviour,
IEEE Trans. on Computers, June 1972, pp. 592-597.

[10] Bourlard H. and Wellekens C. J.,
Multilayer Perceptron and Automatic Speech Recognition,
In proceeding of ICNN-87, pages IV407-IV406,
International Conference on Neural Networks, June 1987.

[11] Chapman D.,
Planning for Conjunctive Goals,
Artificial Intelligence 32 (1987), pp. 333-377.

[12] Cheeseman P.,
In Defense of Probability,
IJCAI 1985 pp. 1002,1009.

[13] Cole R. A., Stern R. M., Phillips M. S., Brill S. M., Specker P., and Pilant A. P.,
Feature-based Speaker-independent Recognition of English Letters,
Proceedings IEEE ICASSP 83, pp. 731-734, 1983.

[14] Cosi P., Bengio Y., and De Mori R.,
On the Generalization Power of Multi-Layered Networks used for Speech Coding,
McGill University Technical Reports, October 1988.

[15] Cravero M., Pieraccini R. and Raineri F.,
Definition and Evaluation of Phonetic Units for Speech Recognition by Hidden Markov Models,
In ICASSP '86 Proceedings, pp.2235-2238. IEEE, 1986.

[16] De Kleers J.,
An Assumption Based TMS,
Artificial Intelligence 28 (1986) 127-162.

[17] De Mori R.,
Computer Models of Speech Using Fuzzy Algorithms, Plenum Press,
New-York, New-York, 1983.

[18] De Mori R., Cardin R., Merlo E., Palakal M. J., and Rouat J.,
A Network of Actions for Automatic Speech Recognition,
Speech Communications special issue on Word Recognition in Large Vocabularies,
Ed. by H. Niemann, vol. 7, no. 4, pp. 337-353, Dec 1988.

[19] De Mori R., Gubrynowicz R., and Laface P.,
Inference of a Knowledge Source for the Recognition of Nasals in Continuous Speech,
IEEE Trans. on Acoust., Speech, Signal Processing,
vol. ASSP-27, no. 5, pp. 538-549, October 1979.

[20] De Mori R., Laface P., and Wong Y.,
Parallel Algorithms for Syllable Recognition in Continuous Speech.
IEEE Trans. Pattern Anal. Machine Intell.,

[21] De Mori R., Lam L., Gilloux M.,
Learning and Plan Refinement in a Knowledge-based System for Automatic Speech Recognition,
IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-9, pp. 289-305, March 1987, Vol. PAMI-7, pp. 56-69, 1985.

[22] De Mori R., Merlo E., Palakal M., Rouat J.,
Use of Procedural Knowledge For Automatic Speech Recognition,
Proc. tenth International Joint Conference on Artificial Intelligence, pp.840-843,
Milan, Italy, August 1987.

[23] De Mori R., Merlo E., and Palakal M.,
Use of Markov Models on Spectral Lines,
submitted for publication.

[24] Demichelis P., De Mori R., Laface P., and O'Kane M.,
Computer Recognition of Plosive Sounds Using Contextual Information.
IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-31, pp. 359-377, 1983.

[25] Denes P. B.,
On the Statistics of Spoken English,
In The Journal of the Acoustical Society of America,
35(6):892-904, June, 1963.

[26] Devijver P. A.,
Baum's Forward-Backward Algorithm Revisited,
Pattern Recognition Letters 3 (1985), December 1985, pp. 369-373.

[27] Doyle J.,
A Truth Maintenance System,

Artificial Intelligence 12 (1979) 231-272.

[28] Ephraim Y., Rabiner L.,
On the Relations Between Modeling Approaches for Information Sources,
Proc. ICASSP 1988, pp. 24-27, IEEE 1988.

[29] Etherington D. W.,
Formalizing Nonmonotonic Reasoning Systems,
Artificial Intelligence 31 (1987) 41-85.

[30] Fahlman S. E., Touretzky, D. S. and Van Roggen, W..
Cancellation in a Parallel Semantic Network,
Proceeding IJCAI 1981, Vancouver, BC(1981) 257-263.

[31] Ferguson J. D.,
Variable Duration Models for Speech,
in Proc. Symp. on Application of Hidden Markov Models to Text and Speech,
J. D. Ferguson, Ed., Princeton, NJ, pp. 143-179, 1980.

[32] Fu K. S.,
Syntactic Pattern Recognition and Applications ,
Prentice Hall, 1982.

[33] Georgeff M. P. and Lansky A. L.,
Procedural Knowledge,
Proc. of IEEE, Special Issue on Knowledge Representation, 1986.

[34] Gonzalez R. G. and T ..nason M. G.,
Synctactic Pattern Recognition - An Introduction,
Addison-Wesley Publishing Company, 1978.

[35] Gupta V. N., Lennig M. and Mermelstein P.,
Integration of Acoustic Information in a Large Vocabulary Word Recognizer,
In ICASSP '87 Proceedings, pp.697-700. IEEE, 1987.

[36] Haralick R. M.,
Personal Communication.

[37] Hatazaki K. and Watanabe T.,
Large Vocabulary Word Detection by Searching a Tree-Structural Word Dictionary,
In ICASSP '87 Proceedings, pp.848-851. IEEE, 1987.

[38] Hinton G. E. and Sejnowski T.J.,
Learning and Relearning in Boltzmann Machines,
In Parallel Distributed Processing : Exploration in the Microstructure of Cognition,
volume 1, pages 282-317, MIT Press, Cambridge, Massachusetts, 1986.

[39] Hopcroft J. E., Ullman J. D.,
Introduction to Automata Theory, Languages and Computation.
Addison-Wesley Publishing Company, 1979.

[40] Huang X., Cai L., Fang D., Ci B., Zhou L. and Jian L.,
A Large-Vocabulary Chinese Speech Recognition System,
In ICASSP '87 Proceedings, pp.1167-1170. IEEE, 1987.

[41] Jelinek F.,
Continuous Speech Recognition by Statistical Methods,
Proceeding of the IEEE, Vol. 64, No. 4, pp. 532-557, April 1976.

[42] Jelinek F.,
The Development of an Experimental Discrete Dictation Recognizer,
IEEE Proceedings, pages.1616-1624, November 1984.

[43] Jelinek F., Averbuch A., Bahl L., Bakis R., Brown P., Daggett G., Das S., Davies K., De Gennaro S., De Souza P., Epstein E., Fraleigh D., Lewis B., Mercer R., Moorhead J., Nadas A., Nahamoo D., Picheny M., Shichman G., Spinelli P., Van Compernolle D. and Wilkens H.,
Experiments with the Tangora 20,000 Word Speech Recognizer,
In ICASSP '87 Proceedings, pp.701-704. IEEE, 1987.

[44] Juang B., Rabiner L. R.,
Mixture Autoregressive Hidden MArkov Models for Speech Signals,
IEEE Trans. on Acoustic, Speech and Signal Processing, Vol. ASSP-33, No. 6, Dec 1985.

[45] Kimball O., Cosell L., Schwartz R., Krasner M.,
Efficient Implementation of Continuous Speech Recognition on a Large Scale Processor,

in Proc. Int. Conference on Acoustics, Speech and Signal Processing, Dallas, TX., pp. 852-855, 1987.

[46] Kister K. F.,
Dictionary Buying Guide, R.R.Bowker Company, New York & London, p.20, 1977.

[47] Klatt D. H.,
Review of the ARPA Speech Understanding Project,
J. Acoust. Soc. Amer., vol. 62, pp. 1345-1366, 1977.

[48] Kopec G.,
Formant Tracking Using Hidden Markov Models,
Proceedings of the International Conference on Acoustics. Speech and Signal Processing, pp. 1113-1116, 1985.

[49] Kopec G., and Bush M.,
Network-based Isolated Digit Recognition Using Vector Quantization,
IEEE Transactions on Acoustics, Speech and Signal Processing,
Vol. ASSP-33, pp. 850-856, August 1985.

[50] Laface P., Micca G. and Pieraccini R.,
Experimental Results on a Large Lexicon Access Task,
In ICASSP '87 Proceedings, pp.809-812. IEEE, 1987.

[51] Lee K. F.,
Incremental Network Generation in Word Recognition.
Proceedings IEEE ASSP Int. Conference pp. 77-80, Tokyo, Japan, April 1986.

[52] Lee K. F.,
Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The Sphinx System,
Ph. D. Thesis, Carnegie Mellon University, 1988.

[53] Levinson S. E.,
Structural Methods in Automatic Speech Recognition,
Proceedings of the IEEE, Vol. 73, No. 11, pp. 1625-1650, 1985.

[54] Levinson S. E., Rabiner L. R., and Sondhi M. M.,
An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition,

Bell Syst. Tech. J., Vol. 62, pp. 1035-1074, 1983.

[55] Lippman R. P.,
An Introduction to Computing with Neural Nets,
IEEE ASSP Magazine, V. 4, n. 2, pp. 4-22, April 1987.

[56] Madhavji N.,
Software Engineering,
course notes, 1987.

[57] Mathan L. S.,
Speaker-Independent Access to a Large Lexicon,
McGill University, Master Thesis, July 1987.

[58] McCarthy, J. and Hayes P. J.,
Some Philosophical Problems from the Standpoint of Artificial Intelligence,
in B. Meltzer and D. Michie (Eds.), Machine Intelligence 4 (American Elsevier, New York, 1969), pp. 463-502.

[59] McDermott D., Doyle J.,
Nonmonotonic Reasoning I,
Artificial Intelligence 13 (1980) 42-72.

[60] Merialdo B.,
Speech Recognition with Very Large Size Dictionary,
In ICASSP '87 Proceedings, pp.364-367. IEEE, 1987.

[61] Merlo E.,
Use of Procedural Knowledge for Spoken Letters and Digits Recognition,
in Recent Advances in Speech Understanding and Dialog Systems,
Ed. by H. Niemann, M. Lang, and G. Sagerer,
Springer-Verlag, 1988.

[62] Merlo E., De Mori R., Palakal M. and Mercier G.,
A Continuous Parameter and Frequency Domain Based Markov Model,
in Proc. Inter. Conf. on Acoust., Speech, Signal Processing,
pp. 1597-1600, Tokyo, Japan, 1986.

[63] Naccache N. J. and Shinghal R.,
SPTA: A Froposed Algorithm for Thinning Binary Patterns,
IEEE Transactions on Systems, Man and Cybernetics,
Vol. SMC-14, No. 3, pp. 409-419, 1984.

[64] Ney H.,
The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition,
IEEE Trans. on Acoustic, Speech, and Signal Processing,
Vol. 32, No. 2, April 1984.

[65] Ni H. P., Feigeinbaum E. A., Anton J. J., Rockmore A. J.,
Signal-to-symbol Transformation. HASP/SIAP Case Study,
The Artificial Intelligence Magazine,
vol. 3, No. 2, pp. 23-35, 1982.

[66] Nilson N. J.,
Principles of Artificial Intelligence,
Tioga Publishing Company, Palo Alto, California, 1980.

[67] O'Shaughnessy D.,
A Tutorial on the Technical Aspects of Synthetic Speech Devices - their Applications and Limitations,
In IEEE Communications Magazine, pp.32-33, December 1983.

[68] Palakal M. J.,
Morphological Representation of Speech Knowledge for Automatic Speech Recognition Systems,
Concordia University, Montreal, Quebec, Ph. D. Thesis. 1987.

[69] Pednault E. D. P.,
Toward a Mathematical Theory of Plan Synthesis,
Ph. D. thesis, forthcoming,
Department of Electrical Engineering, Stanford University, Stanford California.

[70] Plout D. C. and Hinton G. E.,
Learning Sets of Filters Using Back Propagation,
Computer Speech and Language,
2(2):35-61, July 1987.

[71] Poriz A. B.,
Hidden Markov Models: A Guided Tour,
Proc. ICASSP 1988, pp. 24-27, IEEE 1988.

[72] Rabiner L. R., Levinson S. E., and Sondhi M. M.,
On the Application of Vector Quantization and Hidden Markov Models to

Speaker-Independent, Isolated Word Recognition,
The Bell System Technical Journal, Vol. 62, No. 4, April 1983.

[73] Reiter R.,
A Logic for Default Reasoning,
Artificial Intelligence 13 (1980) 81-132.

[74] Rich E.,
Artificial Intelligence,
1983, McGraw-Hill Inc.

[75] Rouat J.,
Analyse et Reconnaissance des Consonnes Nasales, Liquide et Fricatives Indépendamment du Locuteur pou Intégration dans un Système de Reconnaissance Automatique de la Parole Alliant Approches Cognitive et Statistique,
Sherbrooke University, Sherbrooke, Québec, Ph. D. Thesis, 1988.

[76] Rudnicky A. I., Baumeister L. K., De Graaf K. H. and Lehmann E.,
The Lexical Access Component of the CMU Continuous Speech Recognition System,
In ICASSP '87 Proceedings, pp.376-379. IEEE, 1987.

[77] Rumelhart D. E., Hinton G. E. and Williams R. J.,
Learning Internal Representation by Error Propagation,
In Parallel Distributed Processing : Exploration in the Microstructure of Cognition, volume 1,
pages 318-362, MIT Press, Cambridge, Massachusetts, 1986.

[78] Rumelhart D. E., and McClelland L.,
Parallel Distributed Processing: Explorations in the Microstructure of Cognition,
MIT Press, 1986.

[79] Shafer G.,
A Mathematical Theory of Evidence,
Princeton University Press, Princeton, NJ, 1976.

[80] Shapiro L. G., McDonald R. M., Sternberg S. R.,
Shape Recognition with Mathematical Morphology,
in Proc. 8th Inter. Conf. on Pattern Recognition,
IEEE C.N. 86 CH 2342-4, Paris, France, pp. 416-418, 1986.

[81] Shipman D. W. and Zue V. W.,
Properties of Large Lexicons: Implications for Advanced Isolated Word Recognition Systems,
In ICASSP '82 Proceedings, pp.546-549. IEEE, 1982.

[82] Shortliffe E. H. and Buchanan B. G.,
A Model of Inexact Reasoning in Medicine,
Math. Biosci. 23 (1975) 351,379.

[83] Stevens K. N.,
Acoustic Correlates of Some Phonetic Categories,
J. Acoust. Soc. Amer., vol. 68, pp. 836-842, 1980.

[84] Waibel A.,
Prosody and Speech Recognition. Ph.D. Thesis at Carnegie-Mellon University,
CMU-CS-86-162, October 27 1986.

[85] Waibel A., Hanazawa T., Shikano K.,
Phoneme Recognition : Neural Networks vs Hidden Markov Models,
In proceedings ICASSP-88, pages 107-110,
International Conference on Acoustics, Speech and Signal Processing,
April 1988.

[86] Waldinger R.,
Achieving Several Goals Simultaneously,
in Machine Intelligence, E. Elcock and D. Michie eds.,
Ellis Horwood, pp. 8, 94-136, 1977.

[87] Watrous R. L. and Shastri L.,
Learning Phonetic Features Using Connectionist Networks,
In proceedings IJCAI-87, pages 851-854,
International Joint Conference on Artificial Intelligence, August 1987.

[88] Wellekens C. J.,
Explicit Time Correlation in Hidden Markov Models for Speech Recognition,
Proc. ICASSP 1987, pp. 384-386, IEEE 1987.

[89] Wilkins D. E.,
Domain Independent Planning: representation and plan generation,

Artificial Intelligence, 22:269-301, 1984.

[90] Woods W., A.,
Transition Network Grammars for Natural Language Analysis,
Comm. ACM 13, No. 10, 1970, pp. 591-606.

[91] Zadeh L. A.,
Fuzzy Sets and Applications: Selected Papers by L. A. Zadeh,
Edited by R. R. Yager, S. Ovchinnikov, R. M. Tong, H. T. Nguyen,
John Wiley & Sons, Inc, 1987.

[92] Zue V. W. and Lamel L. F.,
An Expert Spectrogram Reader: A Knowledge-based Approach to Speech Recognition,
IEEE International Conference on Acoustics, Speech and Signal Processing,
Tokyo, pp. 1197-1200, 1986.

[93] Zue V. W.,
The Use of Speech Knowledge in Automatic Speech Recognition,
IEEE Proceedings, pp. 1602-1615, November 1985.

PUBLICATIONS

This thesis has produced the following publications:

[1] Bengio Y., Cardin R., Cosi P., De Mori R., and Merlo E.,
Speech Coding with Multilayer Networks,
in: Neurocomputing - Algorithms, Architectures, and Applications,
Ed. by F. Fogelman, Springer-Verlag, to be published.

[2] Bengio Y., Cardin R., De Mori R., and Merlo E.,
Programmable Execution of Multi-Layered Networks for Automatic Speech Recognition,
Communications of the ACM, Feb 1989.

[3] De Mori R., Cardin R., Merlo E., Palakal M. J., and Rouat J.,
A Network of Actions for Automatic Speech Recognition,
Speech Communications special issue on Word Recognition in Large Vocabularies,
Ed. by H. Niemann, vol. 7, no. 4, pp. 337-353, Dec 1988.

[4] De Mori R., Merlo E., Palakal M., Rouat J.,
Use of Procedural Knowledge For Automatic Speech Recognition,
Proc. tenth International Joint Conference on Artificial Intelligence, pp.840-843,
Milan, Italy, August 1987.

[5] Merlo E.,
Use of Procedural Knowledge for Spoken Letters and Digits Recognition,
in Recent Advances in Speech Understanding and Dialog Systems,
Ed. by H. Niemann, M. Lang, and G. Sagerer,
Springer-Verlag, 1988.

[6] Merlo E., De Mori R., Palakal M. and Mercier G.,
A Continuous Parameter and Frequency Domain Based Markov Model,
in Proc. Inter. Conf. on Acoust., Speech, Signal Processing,
pp. 1597-1600, Tokyo, Japan, 1986.