THE USE OF FORMAL GRAMMARS IN AUTOMATIC SPEECH RECOGNITION

A Thesis submitted to
The Department of Electrical and Electronic Engineering
The University of Aston in Birmingham
for the degree of
Doctor of Philosophy

By

CHAIYAPORN CHIRATHAMJAREE

B.E., M.Sc.

FEBRUARY 1979

THE USE OF FORMAL GRAMMARS IN AUTOMATIC SPEECH RECOGNITION

A Thesis submitted to The Department of Electrical and Electronic Engineering, The University of Aston in Birmingham for the degree of Doctor of Philosophy By CHAIYAPORN CHIRATHAMJAREE B.E., M.Sc.

FEBRUARY 1979

SUMMARY

An automatic isolated-word recognition (IWR) system normally consists of a feature extractor (FE) followed by a recognition processor. Some form of 'training' is usually required in order to combat problems of variations in speech. This thesis presents the application of formal grammars to model a FE in an IWR system. The method is to construct, in the training mode, one grammar for each word in the vocabulary, directly from a set of sample strings of 'features' represented by symbols. In the recognition mode, an incoming string is analysed to determine which grammar, if any, could have generated it.

Inference algorithms for both finite-state grammars (FSG's) and context-free grammars (CFG's) considered here are based on the criterion of maximizing the similarity between various strings of the same word. The classification of a string involves the use of the 'weighted matching network' technique in the FSG approach and the computation of the minimisation matrix M for the CFG approach.

Both the FSG and CFG models offer comparable recognition performances whilst the use of the CFG approach results in an increase in the amount of computation required. It appears, therefore, that there is no advantage gained, in terms of recognition performance and computational requirement, from the use of CFG approach over that of the FSG in the recognition of isolated words.

The use of formal grammar approach over the direct storage of strings in isolated-word application makes possible the 'generalisation' of strings in the training set. This can reduce the number of strings required by the learning process. Another advantage of the linguistic method is the reduction in the amount of computation in the FSG approach which is a result of the merging between similar segments of various strings during the training process.

KEYWORDS: Formal grammars, automatic speech recognition, finite-state grammars, context-free grammars, grammar inference .

ACKNOWLEDGEMENTS

It is the author's pleasure to thank his supervisor, Dr. M. H. Ackroyd, for constant guidance, helpful suggestions and encouragement throughout the duration of the research and for introducing him to the subject of formal languages.

Special thanks are also due to Professor J. E. Flood, and the University of Aston in Birmingham for providing the financial assistance in the form of a University research studentship without which it would not be possible to carry out this work.

LIST OF PRINCIPAL SYMBOLS AND ABBREVIATIONS USED

(Subscripts and/or superscripts may be attached to symbols)

£	the start symbol
&	the terminating node
$\in$	set membership
$\cup$	set union
$\cap$	set intersection
$\longrightarrow$	can be replaced or rewritten by (used in rewriting rules or productions)
$\Longrightarrow$	directly derives
$\overset{*}{\Longrightarrow}$	derives ie. $\overset{i}{\Longrightarrow}$ for some $i \geqslant 0$ where $\overset{i}{\Longrightarrow}$ denotes the i-fold product of $\Longrightarrow$
$\overset{+}{\Longrightarrow}$	derives in a nontrivial way ie. $\overset{i}{\Longrightarrow}$ for some $i \geqslant 1$
ASR	automatic speech recognition
AWSL	average weighted string length
A,B,C	nonterminals
A_{H_i}	nonterminal with hierarchy level i
A_{b_i}	nonterminal corresponding to b_i in a terminating rule $A_{b_i} \longrightarrow b_i$
a,b	terminals
$\alpha, \beta, \gamma, \delta$	strings of nonterminals and terminals
b_j	the jth symbol of s
b_{j_i}	the ith symbol of s_j in S_+
b_{j_ℓ}	the last symbol of s_j in S_+ where ℓ is the length of s_j
CF	context-free
CFG	context-free grammar
CFL	context-free language
CS	context-sensitive
CSG	context-sensitive grammar
CSL	context-sensitive language

D_ℓ	minimum value of $d_\ell(k)$ for all values of k
D_m	minimum value of $d_m(k)$ for all words in the vocabulary
$d_\ell(k)$	absolute value of the difference between WSL of a string s and AWSL associated with word w_k
$d_m(k)$	WLD between a string s and the CFG associated with word w_k
e(i,j)	element in row i and column j of the WMN
FE	feature extractor
FSG	finite-state grammar
FSL	finite-state language
FTN	finite-state transition network
$F_D(j)$	non-negative deletion function
$F_I(k)$	non-negative insertion function
$F_S(jk)$	non-negative substitution function
F_j	number of string s_j
f_j	estimated probability of s_j in S_+
GI	grammar inference
G	a phrase-structure grammar or just grammar
G_1	grammar constructed from the first string in S_+ ie. the SG
G_{n-1}	the (n-1)th inferred grammar
G_n	the nth inferred grammar
G_{ns}	SNCFG
G_s	SFSG
HL	hierarchy level (of a nonterminal)
H(k)	hierarchy level of a nonterminal A_k
IWR	isolated-word recognition
$I(s_j)$	number of steps in the derivation of s_j
i	symbol index
J	number of distinctly different derivations of a string y_k in word w_k
j	string index

j_i	the j index of m_{ijk} corresponding to a substring of length i
K	optimal path number
k	word index, row index of WMN
LD	Levenshtein distance
LPC	linear predictive coding
LS	left side (of a rule)
L(G)	language generated by G
L_m	length of the longest member in any rewriting rule of G
ℓ	string length
λ	the empty or null string
MLC	maximum-likelihood criterion
M	minimisation matrix
M_D	number of distinct strings in S_+
M_S	number of strings in S_+
M(k)	number of strings in word k
m	number of nodes in FTN
m_{ijk}	an element of M matrix
NE	nondeterministic event
$N_{ik}(s_j)$	number of times that production $A_i \longrightarrow \alpha_k$ is used in the derivation of s_j
n_{ik}	expected (estimated) number of times that rule $A_i \longrightarrow \alpha_k$ is used in parsing all strings in S_+
PTN	push-down transition network
P_k	set of ordered pairs (p,q) such that $A_k \longrightarrow A_p A_q$ is a rule of a CFG
P_K^k	product of any combinations of $P_D(a_{ij})_k$, $P_I(b)_k$, and $P_S(a_{ij})_k$
P_T^k	summation of P_K^k for all optimal path number K

$P(s_j)$ probability of s_j

$P(w_k)$ the a priori probability of word w_k

$P(s_j/w_k)$, $Q(k)$ product of all production probabilities corresponding to word w_k whose rules are used in parsing s_j

$P(w_k/s_j)$ probability that s_j is in word w_k

$P_D(a_{ij})_k$ probability that a_{ij} is deleted from a string in word w_k

$P_I(b)_k$ probability that b is inserted into a string in word w_k

$P_S(a_{ij})_k$ probability that a_{ij} is substituted by a symbol b

p_{ij}, p_{ik}, p_{kj} production probability

$p(b)$ probability of symbol b

$p(a_{ij}/w_k)$ probability of a_{ij} given that it is in word w_k

$p(b/w_k)$ probability of b given that it is in word w_k

$p(w_k/a_{ij})$ probability that a_{ij} is in word w_k

$p_i(s_j)$ probability of the production used at the ith step of the derivation of s_j

Q_m maximum value of $Q(k)$ for all values of k

RS right side (of a rule)

R a finite set of productions or rewriting rules

R_{ns} rules of SNCFG

R_s rules of SFSG

$\|R\|$ number of rules (total) in a grammar

$\|R_B\|$ number of bielement rules (not including start rules) in a CFG

$\|R_T\|$ number of terminating rules in a CFG

$\|R_{St}\|$ number of start rules in a CFG

r number of nonterminals in a Chomsky normal form CFG

SCF-B stochastic context-free B (recognition system)

SFSG stochastic finite-state grammar

SFS-A	stochastic finite-state A (recognition system)
SFS-B	stochastic finite-state B (recognition system)
SG	skeleton grammar
SNCFG	stochastic normal form context-free grammar
STM-B	stochastic template matching B (recognition system)
S_+	'positive information' sample set
s,x,y	a string of symbols
s_j	the jth string in S_+
s_n	the nth observed string
T	a transition matrix
t_D	deletion coefficient
t_I	insertion coefficient
t_S	substitution coefficient
t(i,j)	element of T matrix associated with a rule $A_i \longrightarrow a_{ij}A_j$ of a FSG
Σ	an alphabet or a finite set of terminals
Σ^*	the set of all strings, including λ, consisting of symbols from Σ
Σ^+	the set of all strings in Σ^* excluding λ
$\|\Sigma\|$	number of terminals of a grammar
V	a finite set of nonterminals and terminals from the union of V_N and Σ
V^*	the set of all strings, including λ, consisting of symbols from V
V^+	the set of all strings in V^* excluding λ
V_{LS}	premise nonterminal (or node)
V_N	a finite set of nonterminals
V_N^*	the set of all strings, including λ, consisting of symbols from V_N

V_P	node nearest to the beginning of a string
V_{RS}	consequence nonterminal (or node)
$\|\|V\|\|$	number of terminals and nonterminals in a grammar
$\|\|V_N\|\|$	number of nonterminals in a grammar
$\emptyset$	the empty set
WHL	weighted hierarchy level
WLD	weighted Levenshtein distance
WMN	weighted matching network
WSL	weighted string length
W	number of words in the vocabulary
W_N	number of words that correspond to string s_j
w_i	one of the words in W_N
wM	weighted minimisation matrix
wH(k)	weighted hierarchy level of A_k
$\|x\|$	length of string x
y_k	a string associated with word w_k
Z	number of links of a FTN
Z_F	number of links of a FTN associated with the FSG approach
Z_T	number of links of a set of FTN's associated with template matching approach

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

Since the early days, man-machine interaction has usually been accomplished by manipulating some mechanical devices such as keyboards, push-buttons, dials, switches etc. This form of communication poses several limitations and drawbacks to the smooth and effective running of machines. For example, it is usually necessary for a human operator to adapt himself to the operational requirements of machines. This requires substantial training of personnels concerned in order to obtain basic physical skill needed for speedy and efficient operations. Also, special preparations of input data, the format of which is governed by the machine concerned, are required before it can be accepted and processed.

A more attractive and better preferred mode of man-machine communication is by means of speech-man's most natural, convenient and basic method of communication. This considerably reduces the disadvantages associated with non-speech man-machine communication systems and offers many desirable features and advantages[1-5], such as the increase in speed of communication, possibilities for mobility and freeing hands and eyes where required, reduction in operating cost etc. In addition, the ability of a machine to respond directly to verbal interrogation fulfils the ultimate aim in communications between man and machine.

Much effort has been put into the research of man-machine communication by speech and recently, several voice input systems, though limited in capability, are available commercially and have been in operation in various fields of applications, some of which

are summarized below:-

(a) Aids for the handicapped[6] (eg. to control bed, lights etc.)

(b) Automated material handling[7-9] (eg. air-line baggage handling, parcel/mail post destination sorting etc.)

(c) Quality control and inspection[10] (eg. inspection of pull-ring can lids, television faceplate, automobile assembly line etc.)

(d) Applications in aircraft[7] (eg. to adjust radio receiving channel etc.)

(e) Applications to computer-based systems[7,10] (eg. parts programming for numerical control of machine tools etc.)

1.2 Systems for the recognition of speech

Basically, an automatic speech recognition (ASR) system is one which can recognize, interpret and respond to speech sound uttered by a human talker. There are several types of ASR machines[5,11] though all of them can be broadly categorized into two groups: continuous and isolated speech recognition systems. In the latter, an isolated-word recognition (IWR) machine included, short pauses are required before and after utterances to be recognized whilst there is no such restriction in the former.

Following the common practice in the field of pattern recognition[12] , an IWR system can be considered as to consist of a feature extractor (FE) followed by a recognizer or classifier as shown in Fig. 1.1 . In this configuration - which is also used by many experimental systems[1,13-21] - when a word is uttered, a decision is made by the recognizer as to which word, if any , in the vocabulary has been spoken. The descriptions for each subsystem follow.

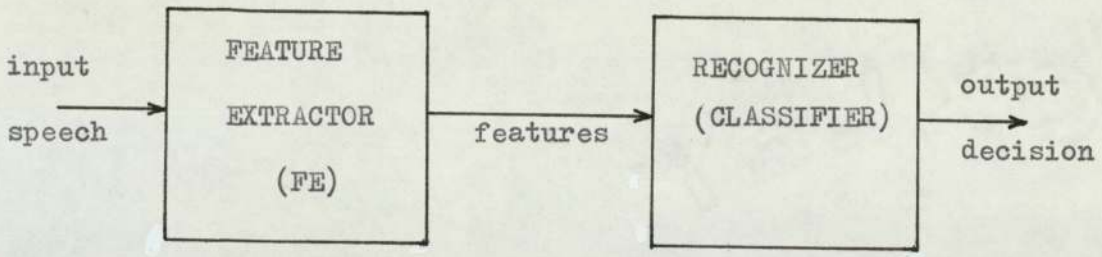


Fig. 1.1 Block diagram of an isolated-word recognition system

1.2.1 Feature extraction

After being converted into electrical energy by a transducer (eg. a microphone or a telephone), the speech wave undergoes the first stage of preprocessing operation designed to enhance the quality of the signal and to reduce the degradation caused by noise. A further process involves the development of procedures for extracting relevant parameters or 'features' from the speech signal. At this stage, some sort of 'data compression' is performed. The aim of a FE is to reduce the data rate of the signal to a manageable level. This is accomplished by discarding irrelevant elements of the signal whilst carefully preserving data which is important and necessary to the recognition of the signal.

Several techniques are employed by various research workers to extract relevant parameters from the speech wave such as spectrum analysis[22], approximation by orthogonal functions, zero-crossing analysis[20,23-25] and linear predictive coding technique (LPC)[26-27]. The speech parameters or features extracted can be presented at the

output of a FE as strings of symbols which are then processed by a recognizer or a classifier and thereby creating a description of the input speech.

1.2.2 Classification

The process of classification can be described as one which identifies the input utterances using the knowledge from strings of features at the output of a FE. The classification methods in many speech recognition systems can be broadly categorized into two main groups.

In the first group, commonly known as the 'template matching' technique or 'pattern matching' method, a set of templates which are the representative patterns or structures of all words in the vocabulary is stored in the system memory. An incoming string from the output of a FE is then compared with each stored template to obtain the 'best match' satisfying some specified criteria[17,19,28-31]. In the second group, the stored rules for constructing strings corresponding to words in the vocabulary are used in the classification of unknown strings[20-21,32-36].

In general, a person does not always speak the same word in the same way. This unconscious alteration of the pronunciation of a word, even spoken by the same talker, may be due to the emotional and physical states of the speaker, the ambient noise level of the surrounding and free variation from trial to trial. Hence, it is important to incorporate some form of 'training' or 'learning' into the speech recognition system, as shown in Fig. 1.2 . In the learning mode, a set of sample strings from the chosen vocabulary is fed into the machine

several times until the representative structures or rules for the construction of each word are formed. This will make the recognition machine able to adapt itself to the characteristics of the talker and thereby reducing the problem of variations in speech.

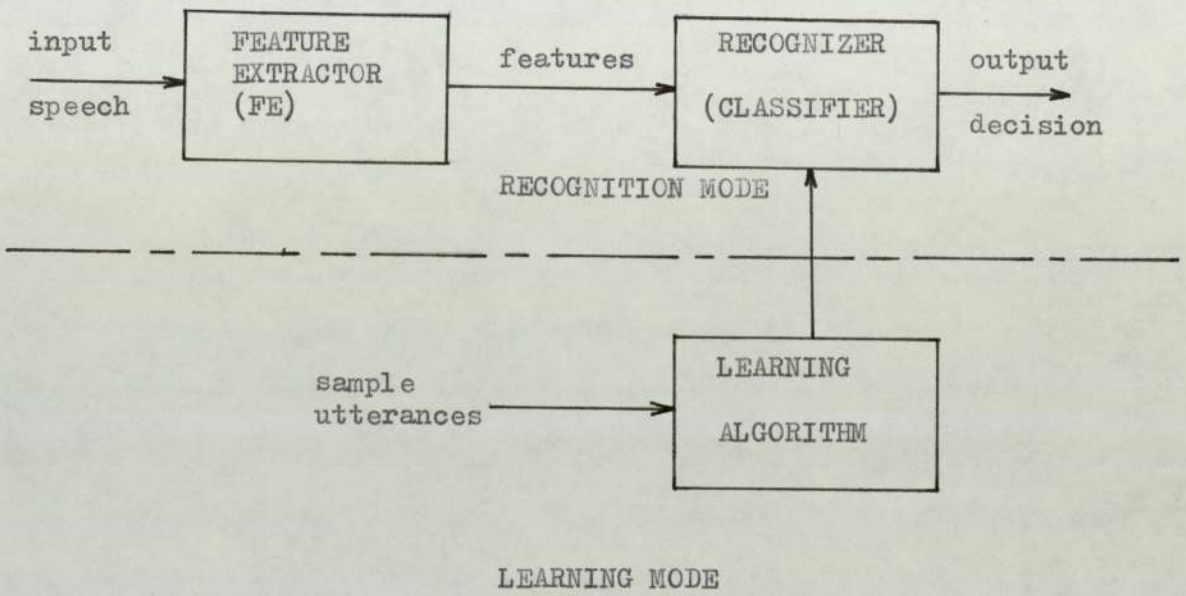


Fig. 1.2 Block diagram of an IWR system with 'learning' facility

1.3 Linguistic approach to ASR

The classical decision-theoretic methods[12,37-39] from the field of pattern recognition, as well as some heuristic methods[20,36] have commonly been used in the past to produce classifiers for processing strings of features from the outputs of FE's in IWR systems.

Another approach, which belongs to the second group of classification methods mentioned in the previous section, is to make use of the technique of formal language theory. This approach, which stemmed from the fields of mathematical linguistics and computer science, has recently received increasing attention[33,40-44] and

provides some promising results in pattern recognition. The essence of the linguistic methods in pattern recognition is to have a grammar for each class of patterns to be recognized. In most cases, a suitable set of grammars are obtained based on a priori knowledge of the characteristics of the patterns together with the experience of the designer of the recognition system under study[45]. In other applications such as ASR, the underlying process of producing patterns may not be clearly understood. In such cases, the only information available, namely, a set of sample patterns is used to construct the required grammars. The search for suitable grammars based on a set of sample strings or patterns is known as 'grammatical inference'[42,48], 'grammar discovery'[46] or 'linguistic learning'[47].

Basically, the linguistic method as applied to ASR works as follows. In the training stage, sets of syntactic rules or grammars are constructed, one for each word in the vocabulary, from a given set of sample strings of features. In the recognition mode, an incoming string from the output of a FE is analysed to determine which grammar, if any, could have generated it. The word corresponding to such grammar is then said to have been recognized.

The 'formmal grammar' approach is sometimes known as the 'syntactic' or 'structural' approach because of the analogy between the hierarchical structure of features or 'patterns' and the syntax of a language. It is attractive to use due to the availability of mathematical linguistics as a tool. In addition, it seems to be well-suited to the problem of an IWR system where only a finite number of features are generated from each utterance. Practical applications of syntactic methods include the design of programming languages, artificial intelligence, information retrieval, scene analysis, chromosome analysis

and many others[49-53].

The work presented in this thesis is concerned with the application of linguistic methods to the design and implementation of classifiers for the recognition of isolated words from a limited vocabulary. It forms part of the research programme on automatic recognition of telephone speech at the Department of Electrical and Electronic Engineering, the University of Aston in Birmingham.

1.4 Outline of the thesis presentation

Chapter 2 introduces preliminary definitions, notation and concepts concerning formal grammars and languages that are related to the present work. Other definitions also appear in subsequent chapters whenever they are required. The second part of this chapter describes the principles involved in the use of formal grammars to model a FE for isolated-word recognition. Basic assumptions and criteria together with some important issues regarding the modelling or the inference process are also given.

Chapters 3 and 4 present the modelling of an IWR system using finite-state grammars (FSG's) and context-free grammars (CFG's) respectively. Methods are given for the construction of a finite-state transition network to graphically represent a FSG and a push-down transition network for a CFG. Inference algorithms and suitable decoding methods for both types of grammars are presented in the appropriate chapters. The FSG approach involves the use of the 'weighted matching network' technique in the recognition process whereas the minimisation matrix M is utilized in both the learning and recognition parts in the CFG approach.

Chapter 5 presents the evaluation and comparison of FSG and CFG models in terms of recognition performance and the computational requirements. Basic recognition systems required for the experimentation are also described. In addition, descriptions are given of the advantages and disadvantages associated with the formal grammar approach and template matching technique.

Chapter 6 presents conclusions and directions for further work which includes the real-time problem and improvements of recognition performances.

CHAPTER 2

FORMAL GRAMMARS AND IWR SYSTEMS

This chapter describes the principles and methods involved in the application of the techniques of formal language theory to the recognition of isolated words. The problem of designing a recognizer in an IWR system can be broadly divided into two areas: the construction of models based on formal grammars to represent the characteristics of the symbol-generating source and the search for suitable decoding methods for efficiently analysing the strings from the source using rules or grammars of the models previously created.

The next section introduces necessary definitions and concepts fundamental to succeeding sections. Other definitions will be given whenever required. For comprehensive treatments of formal grammars see, for example, references 54 and 55 .

2.1 Preliminary definitions, notation and concepts

In IWR systems, words are spoken in isolation with short gaps between utterances. This leads to the following assumptions:-

(i) Only a finite number of features (represented by symbols) are generated by a FE and only one symbol can be present at a particular time.

(ii) Each word uttered results in a sequence of symbols of some finite length.

From the foregoing statements, the following definitions can be made concerning the output of a FE .

Definition 2.1 An alphabet is a set of any finite number of symbols from the output of a FE representing various parameters or features extracted from the input speech wave.

Definition 2.2 A string is any sequence of finite length composed of symbols from the alphabet. The beginning and end of each string are well-defined. A string which contains no symbols is the empty(or null) string λ . The length of a string x denoted by $|x|$, is the number of symbols contained in x .

Definition 2.3 If Σ is an alphabet, then Σ^* denotes the set of all strings consisting of symbols from Σ, including the empty string λ . Also define Σ^+ as $\Sigma^* - \{\lambda\}$.

Developments of the theory of formal languages started when Chomsky first formulated the concept of the hierarchical structures in grammars in 1956[56]. Basically, a grammar or a set of rules can be described as a mathematical system for defining a language, as well as a device for giving a useful structure to the strings in the language.

Formally, a grammar is defined as follows .

Definition 2.4 A phrase-structure grammar G [54-55], or Chomsky grammar[57-58] is defined as :-

$$G = (V_N, \Sigma, R, \pounds) \qquad (2.1)$$

where

V_N is a finite set of nonterminals .

Σ is a finite set of terminals .

$V_N \cap \Sigma = \emptyset$ (the empty set) .

$V_N \cup \Sigma$ is denoted by V .

The terminals in Σ consist of all symbols from the alphabet. All other symbols are nonterminals which rank higher than the terminals in the hierarchical structure of the grammar.

R is a finite set of productions or rewriting rules of the form

$\alpha \rightarrow \beta$ (This implies that α can be replaced or rewritten by β)

where α is a string in V^+

β is a string in V^* .

£ is the start symbol . It is in V_N and signifies the beginning of a string or a word .

Note A rewriting rule with the start symbol £ at the left side (LS) of the rule is known as the start rewriting rule or the start production . For example, £ → aA, £ → bAC, and £ → AB are all start rewriting rules .

Before going on to describe various types of grammars, conventions are given regarding the different types of letters or characters representing terminals and nonterminals .

CONVENTIONS

(a) NONTERMINALS : Capital LATIN - alphabet letters .

(b) TERMINALS : Lower case letters at the beginning of the LATIN alphabet.

(c) STRINGS OF TERMINALS : Lower case letters near the end of the LATIN alphabet .

(d) STRINGS OF NONTERMINALS AND TERMINALS : Lower case GREEK letters.

Grammars can be classified according to the format of their rewriting rules .

Definition 2.5 (54-59) Let $G = (V_N, \Sigma, R, £)$ be a grammar .

The grammar defined in definition 2.4 is a type 0 or unrestricted grammar .

G is said to be :-

(a) type 1 or context-sensitive (CS)

if each production in R is of the form $\gamma A \delta \rightarrow \gamma \beta \delta$

where A is in V_N

γ and δ are in V_N^*

β is in V^+

ie. 'A' is rewritten as 'β' only in the context of $\gamma \ldots \delta$.

(b) type 2 or context-free (CF)

if each production in R is of the form $A \rightarrow \beta$

where A is in V_N

β is in V^+

ie. the rewriting is done independently of the context .

(c) type 3 or regular or right-linear or K-grammar (Kleen's grammar)[60]

if each production in R is of the form $A \rightarrow aB$ or $A \rightarrow a$

where A and B are in V_N

a is in Σ .

A regular grammar is also known as a finite-state grammar (FSG) . This is because the FSG corresponds to a machine with a finite number of states. The application of a rewriting rule is represented by a transition from a state corresponding to the nonterminal at the left-side of the rule to a state corresponding to that at the right .

A grammar with a higher type number is included in the one with a lower type number as shown in Fig. 2.1 .

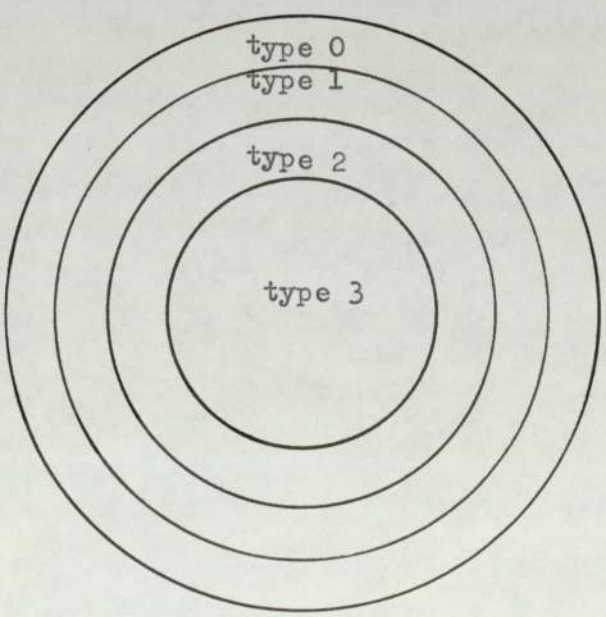


Fig. 2.1 A simplified representation of relationships between various types of grammars

The formal definition of 'language' is given next.

Definition 2.6 The language generated by a grammar G, denoted by L(G) is defined as:- $L(G) = \{x \mid x \text{ in } \Sigma^* \text{ and } \& \overset{*}{\underset{G}{\Rightarrow}} x\}$ (2.2)

where $\& \overset{*}{\underset{G}{\Rightarrow}} x$ means the string x can be derived from & in grammar G. In other words, the language of a grammar consists of all strings of terminals, including the null string, that can be obtained by successive applications of the rewriting rules commencing from the start symbol.

The languages derived from a FSG, a context-free grammar (CFG) and a context-sensitive grammar (CSG) are known as a finite-state language (FSL), a context-free language (CFL) and a context-sensitive language (CSL) respectively.

2.2 Grammar-based modelling in word recognition

This section describes, in general, the use of formal grammars in the formulation of the problem of automatic recognition of isolated words from a limited vocabulary. Fig. 2.2 depicts a generalized block diagram of an IWR system based on formal grammar concepts.
First, one aspect of the learning process is formally defined.

Definition 2.7 A supervised learning is one where the labels of strings in the sample set are known beforehand (eg. a teacher or an observer is available) .

In Fig. 2.2, the FE together with its speech input can be considered as a linguistic information source whose output consists of a collection of finite-length sequences of symbols. The decoder and the models constructed during the learning mode make up the recognizer of the IWR system. It is a normal practice to assume a supervised learning. Thus an observer is present during the learning stage.

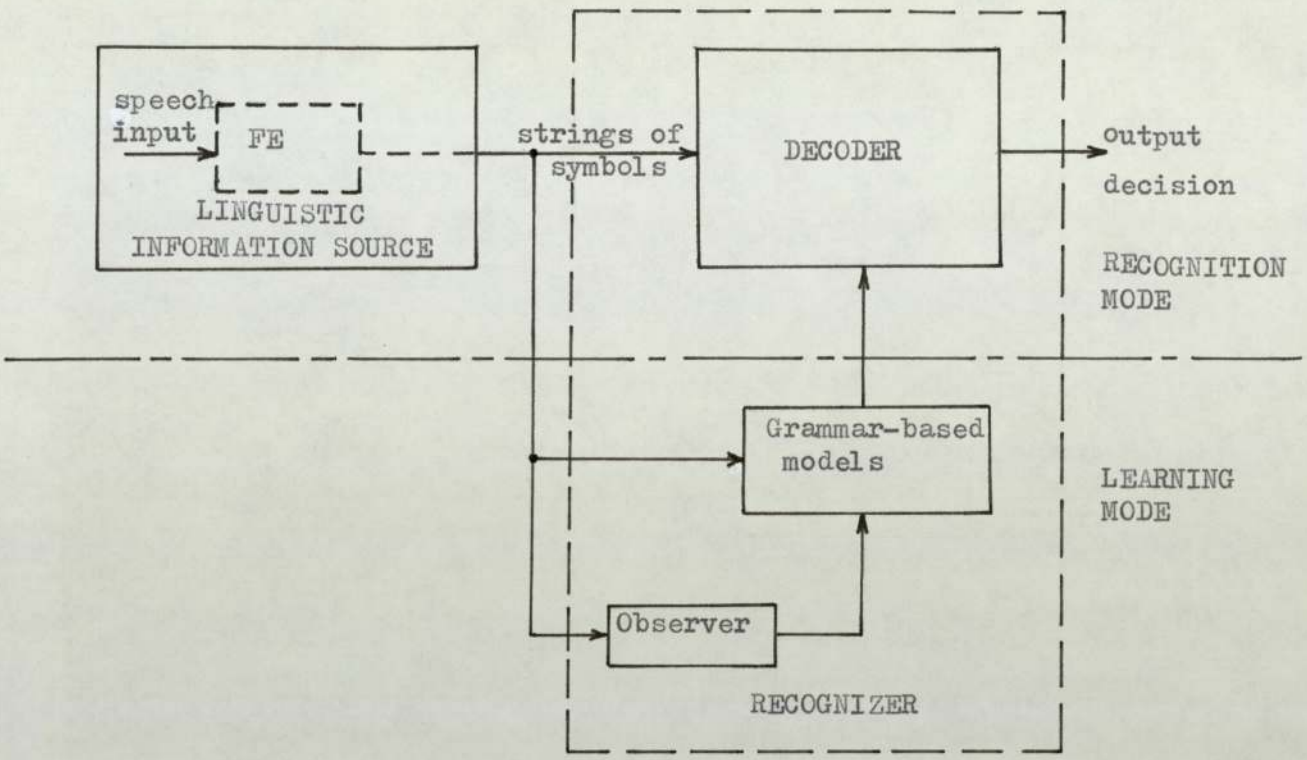


Fig. 2.2 Block diagram of a grammar-based IWR system

Grammar inference (GI) can be viewed as the process whereby formal grammars are employed to model the source whose characteristics are very little, if at all, known. The method is to have, in the training mode, a user repeating each word in the vocabulary a number of times. Each time the same word is spoken, a similar but not necessarily the same string of symbols is produced by the source. Grammar-based models, one for each word in the vocabulary, are then automatically constructed and stored in the system memory for future use. In addition to producing all the strings in the corresponding sample set, each model is also capable of predicting other similar strings. The building of models can also be regarded as a useful encoding of strings.

In the recognition mode, an incoming string is processed using suitable decoding algorithms to determine which model, if any,

corresponds or nearly so to the word spoken. If the most compatible model is found, the corresponding word is then indicated as to have been recognized. Otherwise, the recognition fails and the word is rejected.

The problem of using formal grammars to model a source or GI and its solutions has been studied and its significance stated by various researchers[61-63]. Comprehensive surveys and reviews of previous work and results have also been given[42,44,63,64].

Generally, there are two main approaches to the solution of the problem of learning. They are briefly described below.

(a) Enumerative approach

In this approach, an algorithm is used to produce all grammars of the specified type in an ordered manner. Assuming each class of grammars constructed is denumerable, a method is then established to test these grammars to obtain at least one that meets a given set of criteria.

Although the approach is often shown to give an optimal solution requiring only a minimal amount of information presented, it may be impractical for many applications, for example ASR. This is due to the astronomical amount of computations involved in the exhaustive searching for a suitable grammar. However, the discouragingly enormous amount of combinations involved can be reduced to a certain extent by designing the method such that at any finite time only a finite number of grammars need to be tested.

Some techniques which are inductive as well as probabilistic in nature have been employed in this approach[42,46,61,65,66].

(b) Constructive approach

This method, for example reference 48, constructs one or more grammars directly from the strings in the sample set. The useful, if not optimal grammars based on direct observations of the properties of the strings can be produced in a not too excessive amount of time. Some criteria may also be used to accept or reject the inferred grammars.

Discussions are now made of some important issues concerning the problem of using formal grammars to model the FE in an IWR system.

(1) Data and its structure

Apart from the assumptions about the output of the FE, the only information available during the training mode is a collection of strings of symbols together with their labels. This type of information is known as 'positive information' or 'text-presentation'[61] since only valid strings are known or given. Thus, if probabilistic information is required then it must be estimated from the given sample set. The size of the sample set is arbitrarily specified (eg. five or ten repetitions per word in the vocabulary), though it will be large enough to ensure that the inferred grammar covers a reasonable number of variations of strings representing the same word.

(2) Determination of grammar types

The GI problem is known to be unsolvable for a general (ie. unrestricted) grammar. Thus, many researchers consider the subsets of the general rewriting systems or grammars such as FSG's or CFG's. In many cases of GI, the observed strings from a linguistic source are assumed to have been generated by a precisely defined class of grammars.

In IWR systems or even the general ASR systems, it is not known whether any class of grammar can represent exactly the

characteristics of the ill-defined FE. Indeed, this has been exemplified by Gold[61] who stated that with only positive information available (as in the case of the FE), not even the FSG's are 'identifiable in the limit' - ie. not even the FSG's can be found that will exactly model the FE.

Consequently, the research carried out in this thesis is concerned with finding well-formed approximations to an ill-formed problem (vaguely defined FE) . The types of grammars investigated will be limited to types of up to and including CFG's. The CSG's are not considered here because several problems, such as the closure properties and decidability, are still unsolvable. In addition, the decoding methods for the CSG's are much more complicated than those of the grammars of higher type numbers.

(3) Other criteria

One important requirement in the modelling of a FE by means of formal grammars is that the inferred grammars should be powerful enough to adequately describe data from speech. That is, each grammar should generate all of the known strings (positive information) representing one spoken word in addition to predicting other strings similar in some ways to the observed strings. Ideally, the grammar should also generate none of the known 'non-strings' ie. strings corresponding to other words in the vocabulary.

As mentioned earlier, only positive information is available in IWR systems and hence it is doubtful whether any class of grammars, if any, can describe precisely the nature of the FE. Consequently, speech recognition is a situation where a quick and reasonable inference is more useful than a time-consuming and computationally laborious inference which exhaustively searches for an optimal solution.

It is for these reasons that all the inference algorithms presented in this thesis are mainly constructive in nature.

(4) Basic assumptions

In general, one or more assumptions are normally formed concerning the solution to an inference problem. The following describes some basic assumptions as applied to all the inference methods given in this thesis.

(a) The languages generated by the inferred grammars are assumed to be λ-free. It is meaningless to consider empty strings in any practical application such as ASR where a null string corresponds to no input to the system. This assumption does not restrict the languages in any way.

(b) The generation of grammars in this work is algorithmic in nature to guarantee the convergence of the process.

(c) The inference methods are incremental in the sense that it is possible to update a previously inferred grammar upon receiving a new set of data without the need to store the strings observed earlier ie. there is no need to redo the inference again from the beginning.

(d) The positive information sample set S_+ consists of a finite number of strings each of a finite length. This follows from the assumption about the FE given in section 2.1 .

(e) S_+ is 'structurally complete' [42,65,68] ie. each production in the inferred grammar is used at least once in deriving at least one string in S_+.

CHAPTER 3

FINITE-STATE GRAMMAR BASED MODELLING

There are many approaches to the learning of a specific vocabulary in the recognition of isolated words. The simplest and obvious method is to directly store one or more strings as the representative 'templates'. Another way is by means of 'sequential matching'[32] where an attempt is made to generate sequential structures or lattices from a number of sample utterances. The structures are obtained through successive matching and merging of symbols between strings in the training set. Normally, the criterion used in such processes is to maximize the similarity between the sample strings corresponding to the same word. An alternative approach, for example reference 20, is to construct an algorithm suitable for each word based on observations of sequential characteristics of symbols in each word, allowing alternative and/or optional characters in some positions.

This chapter describes the grammar inference approach to the learning problem based on FSG's. Fig. 3.1 illustrates the general outline of the approach. The jth string in the positive information sample set S_+ is denoted by s_j. The inference algorithm automatically constructs a FSG directly from S_+ on the basis of pre-specified criteria. Suitable decoding techniques are also presented.

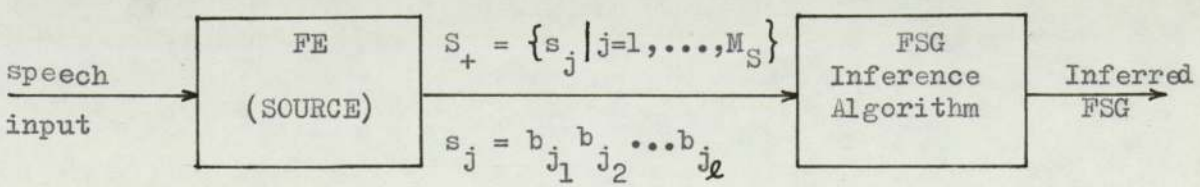


Fig. 3.1 Inference of a FSG in an IWR system

3.1 Graphical representation of a FSG

This section shows how a graph can be applied to portray a FSG such that it is easier to visualize and understand the underlying mechanism of the grammar involved. First, the formal definition of such a graph is given.

Definition 3.1 A finite-state transition network (FTN) is a directed graph having a finite number of nodes or states. A link connecting one node to another indicates the transition originating from the former and terminating in the latter.

A terminal symbol is associated with each and every transition.

Every FSG can be represented by a FTN as follows.

(a) There exist nodes of the FTN corresponding in a one-to-one relationship to nonterminals in V_N of the grammar.

(b) The initial node or the start node of the FTN corresponds to the start symbol £ of the grammar.

(c) A special node called the terminating node & designates the end of a string.

(d) For each production of the form $A \rightarrow aB$, there is a path or transition labelled 'a' from node corresponding to 'A' to node corresponding to 'B'.

(e) For each production of the form $A \rightarrow a$, there is a path labelled 'a' from node corresponding to 'A' to the terminating node & .

Example 3.1 Consider the grammar $G_{3.1} = (V_N, \Sigma, R, £)$ with the following rewriting rules :-

$£ \rightarrow U$ $\quad £ \rightarrow MA'$ $\quad £ \rightarrow VB'$ $\quad £ \rightarrow TH'$

$A' \rightarrow VB'$ $\quad B' \rightarrow AC'$ $\quad C' \rightarrow T$ $\quad H' \rightarrow aI'$ $\quad I' \rightarrow V$

where $V_N = (£, A', B', C', H', I')$

and $\Sigma = (a, A, M, T, U, V)$

The corresponding FTN is depicted in Fig. 3.2 below.

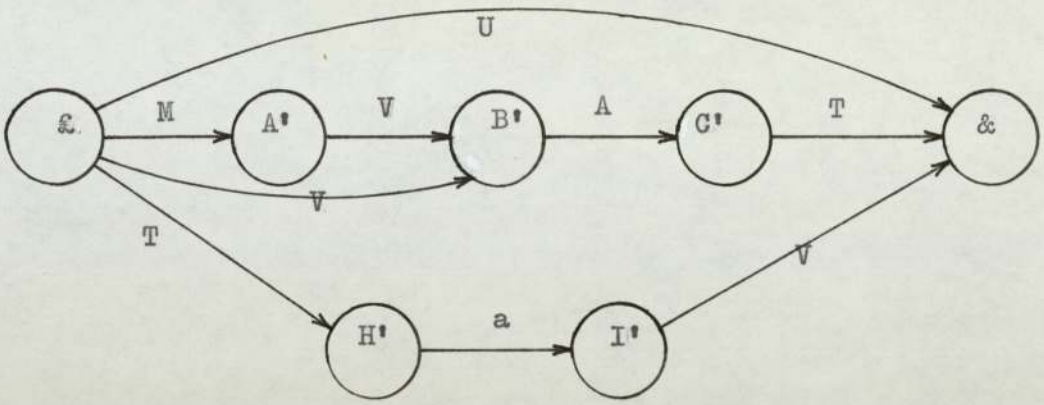


Fig. 3.2 A FTN corresponding to the grammar $G_{3.1}$ in example 3.1

The FTN shown in Fig. 3.2 is similar to Moore's model of sequential machine[67] in the sense that each of the nodes in the FTN, except the initial and terminating nodes, is associated with one and only one symbol. In other words, the symbol produced during a transition corresponds only to the node where the transition ends irrespective of the number of transitions leading to that node. In order to keep the number of nodes (and hence the nonterminals in V_N) small, the terminating node & is allowed to be associated with any number of terminals. The starting node is, of course, associated with none of the symbols since it represents the starting point for all transitions within the FTN.

The inference algorithm described in section 3.2.2 imposes the above constraints upon FTN's in the automatic construction of FSG's.

3.2 Inference of a FSG

The inference of a FSG can be considered as the building of a FTN from a number of sample utterances. Any path that can be traced through the FTN starting from the initial node and ending at the terminating node constitutes the word. It is well known that the relationship between a grammar and the language that it can generate is not unique. That is, there are many grammars that can produce a given language. For example, one grammar may generate exactly those strings in the sample set whilst another may produce not only the strings in the given language but also many other strings. The problem is to find a suitable grammar between these two extremes such that it produces all the strings in the sample set as well as some other strings of similar characteristics.

The following describes some of many advantages and attractive features associated with the use of a FSG in the modelling of a FE in the recognition of isolated words.

(i) A FSG, being the least complicated type of grammars, is simple to construct and test.

(ii) The structure in time of symbol strings from the output of a FE is sequential ie. symbols are assumed to be presented and responded to, at discrete points in time. This resembles very well with the sequential format of a FSG thus rendering the classification problem more attractive to solve.

(iii) The properties and characteristics of a FSG are well established. There exist algorithms to answer many questions such as ambiguity, closure properties and decidability.

(iv) It is easy to read off directly sequences of symbols composing a string corresponding to a word by tracing through the FTN from the initial state to the terminating state.

3.2.1 Criteria for the FSG inference process

Basic assumptions regarding all the inference methods described in this thesis have already been given in section 2.2 . First, some terminology are formally defined in preparation for the presentations to follow.

Definition 3.2 The nondeterministic event (NE) is the situation where an incoming string is assigned by a classifier or a recognizer as corresponding to two or more words in the vocabulary. This indicates the overlapping between strings corresponding to differnt words.

Definition 3.3[55] A FSG $G = (V_N, \Sigma, R, \pounds)$ is said to be recursive if there exists at least one derivation of the form $A \overset{+}{\Longrightarrow} xA$ where $A \in V_N$, $x \in \Sigma^+$ and $\overset{+}{\Longrightarrow}$ implies that the derivation is obtained by the application of one or more rewriting rules. That is, a recursive grammar signifies the occurrence of at least one loop or a closed path in the corresponding FTN.

A method is given in appendix A for testing whether a specified FSG is recursive or not.

The following presents the criteria and related constraints governing the formulation of the FSG inference algorithm to be presented in the next section.

I. The inferred grammar is finite-state.

II. The similarity between strings that can be derived from the inferred grammar should be maximized. This follows from the basic requirement in GI that the grammar created should generate as few non-strings as possible.

III. In the experimental observation of the output of a FE, a training set can consist only of a finite number of finite-length strings. In this situation, it is intuitively felt that a non-recursive FSG

would be appropriate and sufficient to model the FE. Consequently, for each sample string individually considered, no recursive structures are formed or included in the corresponding subset of rules. This also restricts the number of non-strings that the grammar may generate. However, other constraints necessary to the inference process may indirectly give rise to the recursiveness. For example, the addition of new links and/or nodes on the basis of other constraints may produce one or more loops in the FTN. Combining the two requirements above results in the following criterion :-

'Suppress the recursive structure of the inferred grammar as far as is possible but, subject to other constraints, not completely'.

IV. A Moore's model of the FTN is assumed (see section 3.1) . However, two or more nodes (except, of course, the start node) may be associated with the same terminal symbol, though this is kept to a minimum. As an example, in Fig. 3.3 nodes A and D are associated with the same terminal s . This is done to avoid inferring a grammar that is too general ie. one that generates too many non-strings. The above also satisfies criterion III .

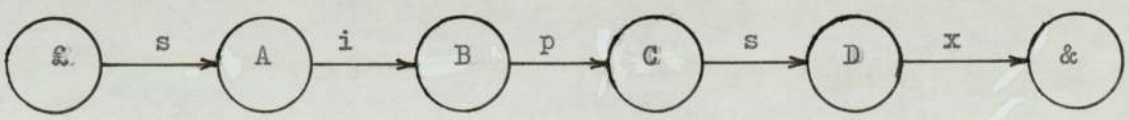


Fig. 3.3 A FTN corresponding to string 'sipsx' with two nodes (A and D) associated with terminal s .

V. In the derivation only of the inferred grammar, each node from the path in the FTN corresponding to a sample string is used once only. For example, the terminal s appears twice (neither of the s's is the last symbol) in the sample string 'sipsx' . Instead of sharing the same node (node A in Fig. 3.4), the two s's are assigned to two

different nodes (A and D) as shown in Fig. 3.3 . Again, this is to satisfy criterion III.

Note also that if the second s happens to be the last symbol of the string, the sharing of the same node by the terminal s would automatically be inhibited. This follows because there can not be any outgoing transition from the terminating node.

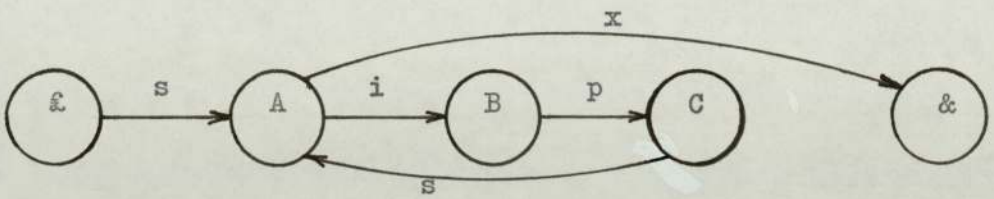


Fig. 3.4 A FTN corresponding to string 'sipsx' with the two s's sharing the same node (node A) resulting in a recursive structure in the grammar.

VI. Tail-end constraint

This is a constraint designed to reduce the occurrence of the NE to the lowest possible level. It determines whether an outgoing link from a current node can be connected to one or more existing nodes corresponding to the next symbol in the string under consideration. The constraint is :-

'Reject the node if neither of the following is satisfied:

(a) The node is a pre-terminating node (ie. the terminating node but one) AND the position of the next symbol is the last symbol but one.

OR

(b) The node is NOT a pre-terminating node AND the position of the next symbol is NOT the last symbol but one. '

VII. Strings of short lengths (≤ 2) are separately dealt with. As an

example, the node corresponding to the first symbol of a length-2 string can only be reached by the start node only. It is also desirable that rules inferred from strings whose lengths are greater than two should not generate any string of length two. This is because strings of short lengths tend to modify and influence the structure of the inferred grammar in such a way that the grammar becomes too general.

It is also of interest to consider the situation where each word in the vocabulary contains exactly one string of unit length in addition to some longer strings. Assuming all length-1 strings are distinct, the maximum number of words that can be correctly recognized is the number of distinct symbols that can be produced by a FE ie. the size of the alphabet. This imposes the limit to the vocabulary size of an IWR system. Such unit length strings are also liable to symbol mutilation since only one alteration is required to corrupt a length-1 string. Fortunately, the foregoing situation rarely happens in any practical application.

The following describes a constraint related to short-length strings.

Front-end constraint

'All nodes corresponding to the nonterminals at the right side (RS) of the start rewriting rules of a FSG cannot be reached by any other nodes except the start node.'

The above will :-

(a) take care of the case of strings of length-2.

(b) ensure that rules inferred from strings whose lengths are greater than two will not generate additional length-2 strings.

(c) keep down the occurrence of the NE.

VIII. If two or more nodes are available for selection, choose the one as near to the beginning of the string concerned as is possible,

provided other constraints are also satisfied. This is to maximize the possibility of branching afterwards from the selected node which results in a greater number of similar strings being produced.

IX. The complexity of the inferred grammar, that is the number of nodes and/or links in the corresponding FTN, should be as near minimum as is possible, subject to other constraints. This implies that the similarity between different strings derivable from the inferred grammar is maximized.

3.2.2 FSG Inference algorithm

A learning algorithm is presented in this section for the automatic generation of a FSG or the corresponding FTN directly from the observed sample strings of sequential features. The method, based on criteria and constraints specified in the previous section, can be briefly explained as follows.

First, 'the skeleton' grammar (SG) G_1 is constructed from the first string in the sample set such that G_1 can generate only that string. Other strings in the sample set are then individually operated upon in the following recursive manner. The nth observed string s_n is analysed with the (n-1)th inferred grammar G_{n-1} to determine whether s_n can be derived from G_{n-1}. If G_{n-1} can generate s_n, then $G_n = G_{n-1}$ and no augmentation of G_{n-1} is required. Otherwise, new rules and/or links are added to the FTN of G_{n-1} to produce the nth inferred grammar G_n.

Before proceeding to present the inference algorithm, some necessary definition and notation are introduced.

Definition 3.4 The nonterminals at the LS and RS of a production of a FSG are known as premise nonterminal and consequence nonterminal

respectively. For example, in the production $A \longrightarrow aB$ the premise and consequence nonterminals are 'A' and 'B' respectively.

Notation

V_{LS} = premise nonterminal (or node)

V_{RS} = consequence nonterminal (or node)

V_P = node nearest to the beginning of a string

ℓ = length of a string

$M(k)$ = number of strings in word k

W = number of words in the vocabulary

s_j = the jth string in the set of a word

b_{j_i} = the ith symbol of s_j

i = symbol index

j = string index

k = word index

The following is algorithm 3.1 which is employed to automatically infer FSG's directly from sample strings of a given set of words in the specified vocabulary. A flow diagram of the inference algorithm is also given in Fig. 3.5 .

Algorithm 3.1

Step 1 Set $k = 0$.

Step 2 Set $k = k + 1$

$j = 0$.

Step 3 Set $j = j+1$.

Read a string $x_j = b_{j_1} b_{j_2} \ldots b_{j_i} \ldots b_{j_\ell}$.

Step 4 Set $V_{LS} = \&$

$i = 1$.

Step 5 Set $V_P = 0$.

(5a) If $i = \ell$, go to step 9.

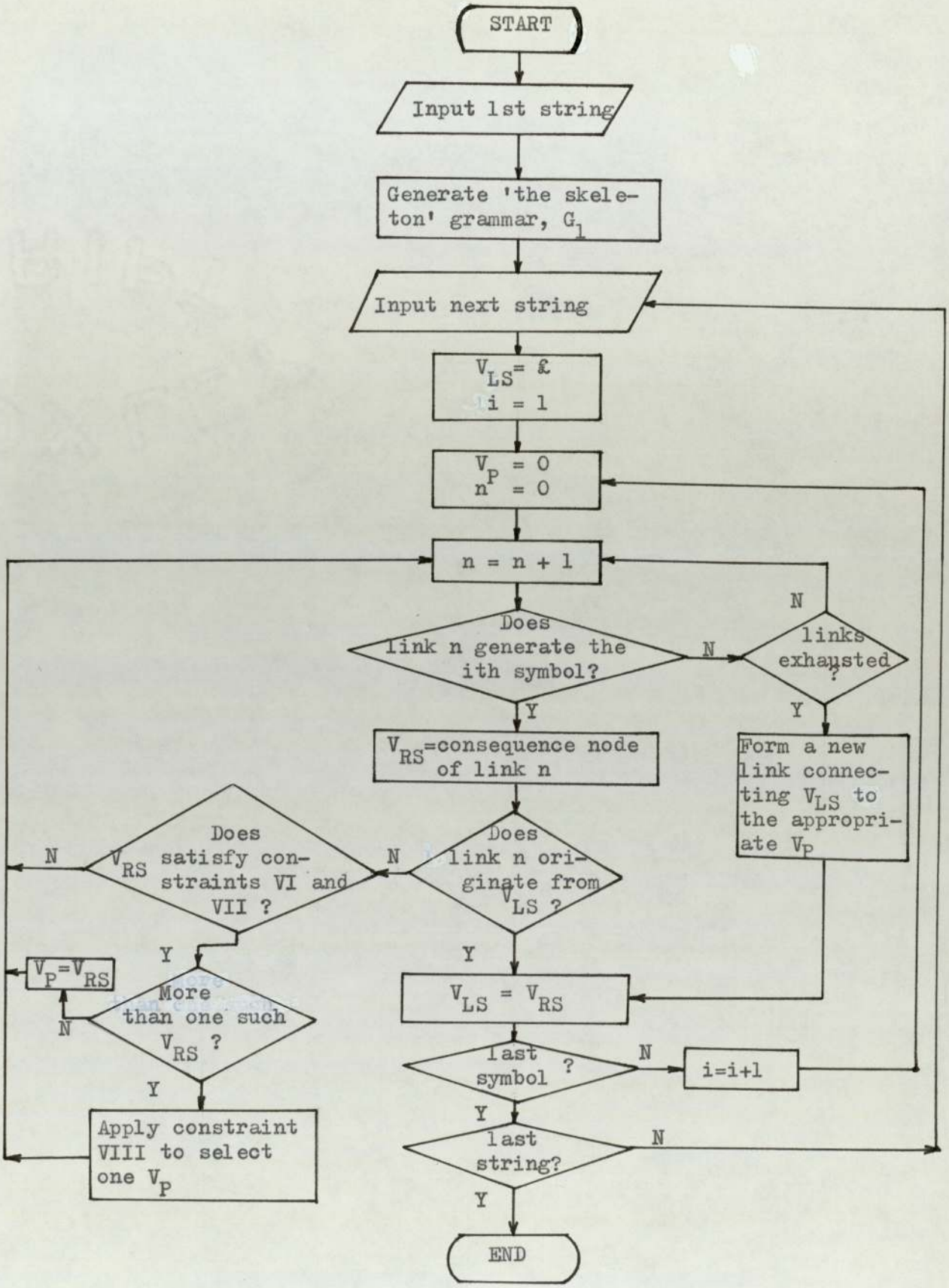


Fig. 3.5 Flow diagram of the algorithm for the inference of a FSG

Compare b_{j_i} with the terminal 'a' in a production of the form $A \longrightarrow aB$, subject to constraint V.

If no such production exists, go to step 6.

(5b) Compare V_{LS} with the nonterminal 'A' .

If V_{LS} = 'A', go to step 7.

Otherwise, go to step 8.

Step 6 Form a new production of the form $A \longrightarrow aB$ and include it in the production set R :-

where 'A' is set to V_{LS}

'a' is set to b_{j_i}

'B' { is set to V_P if $V_p \neq 0$
{ is a new nonterminal if $V_P = 0$.

Step 7 Set V_{LS} = 'B'

i = i + 1 .

Go to step 5 .

Step 8 If 'B' does not satisfy 'front-end' and 'tail-end' constraints, go to step 5a .

If $V_P = 0$, set V_P = 'B' and go to step 5a.

Otherwise, apply constraint VIII to select one value of V_P.

Go to step 5a.

Step 9

(9a) Find a production of the form $A \longrightarrow a$ having 'A' and 'a' identical to V_{LS} and b_{j_ℓ} respectively.

If no such production exists, go to step 9b.

Otherwise, go to step 9c.

(9b) Form a new production of the form $A \longrightarrow a$ and include it in the production set R :-

where 'A' is set to V_{LS}

'a' is set to b_{j_ℓ} .

(9c) If k = W and j = M(k) end .

if j = M(k), go to step 2.

Otherwise, go to step 3 .

Although the skeleton grammar G_1 is depicted explicitly in Fig. 3.5, no special routine is required to generate it. This is because algorithm 3.1 is formulated in such a way as to automatically include the creation of G_1. It is also of interest to note that the SG G_1 is similar to the canonical grammar as defined by FU and BOOTH[42] in the sense that they both generate exactly those strings in the sample set. In the case of G_1 there is, of course, only one string that it can generate. The FSG's inferred by algorithm 3.1, for example the grammars in the next section, can also be viewed as one form of derived grammars[42].

3.2.3 Illustrative example

Example 3.2 Consider a sample set $S_{3.2} = (s_i \mid i = 1,2,\ldots 7)$

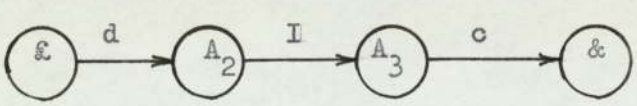
where s_1 = dIc s_2 = eKf s_3 = bJcDi

s_4 = Jg s_5 = bJdCg s_6 = MeCh

s_7 = bKj .

The following illustrates the resulting FSG G_i (shown in the form of FTN) after each string s_i has been presented to algorithm 3.1. The strings that can be derived from each grammar G_i are also given.

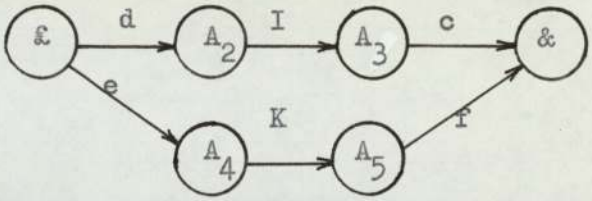
s_1 = dIc



Strings generated by G_1 : dIc

Fig. 3.6 FTN corresponding to G_1, the skeleton grammar

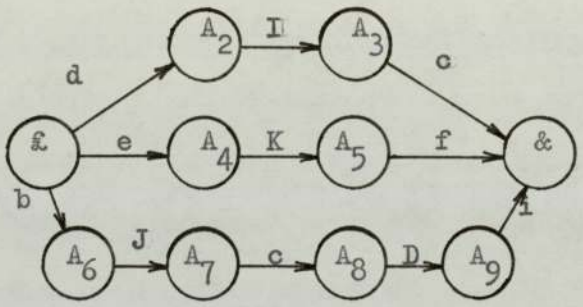
s_2=eKf



Strings generated by G_2 : dIc
eKf

Fig. 3.7 FTN corresponding to G_2

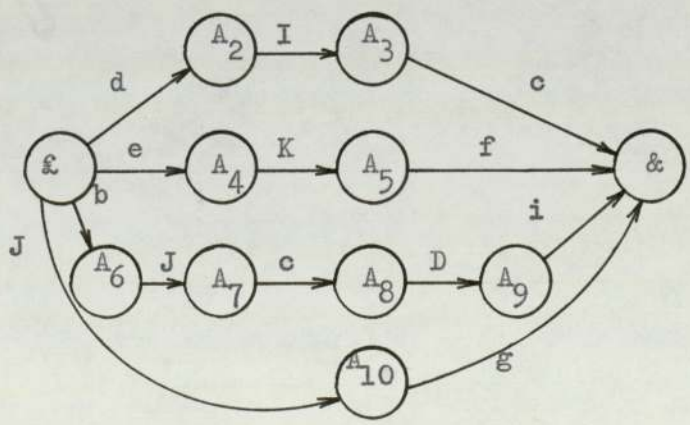
s_3=bJcDi



Strings generated by G_3 : dIc
eKf
bJcDi

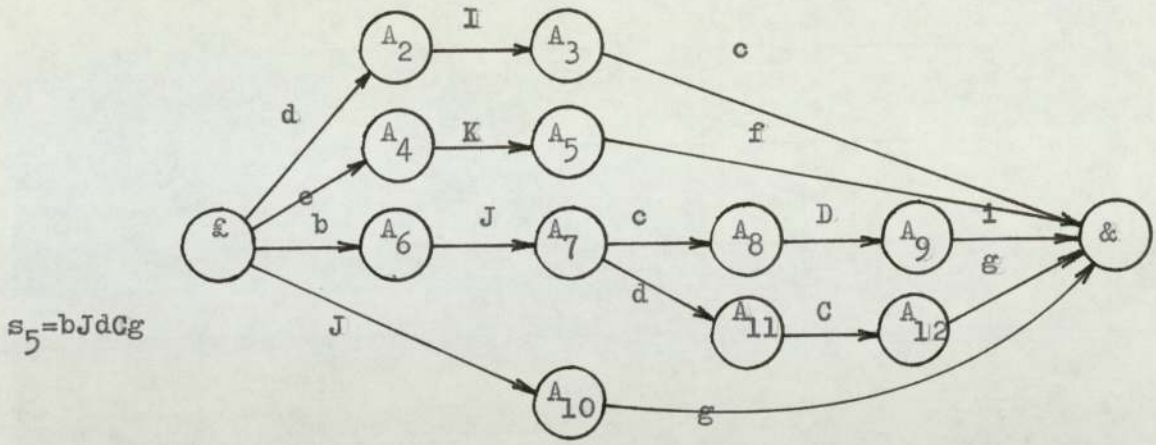
Fig. 3.8 FTN corresponding to G_3

s_4=Jg



strings generated by G_4 : dIc
eKf
bJcDi
Jg

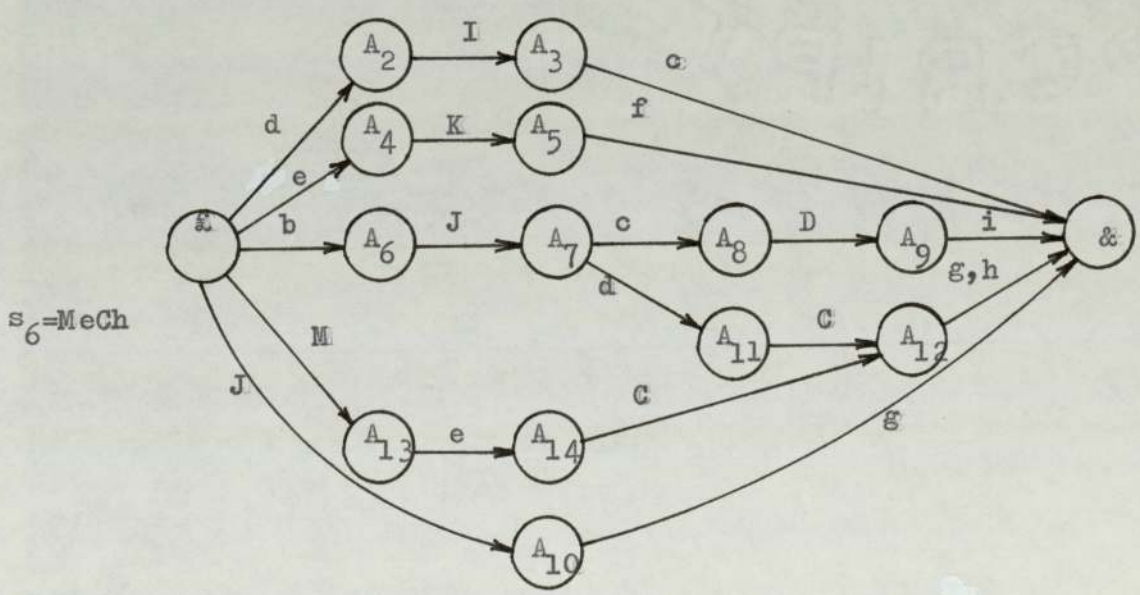
Fig. 3.9 FTN corresponding to G_4



s_5=bJdCg

Strings generated by G_5 : dIc
eKf
bJcDi
Jg
bJdCg

Fig. 3.10 FTN corresponding to G_5

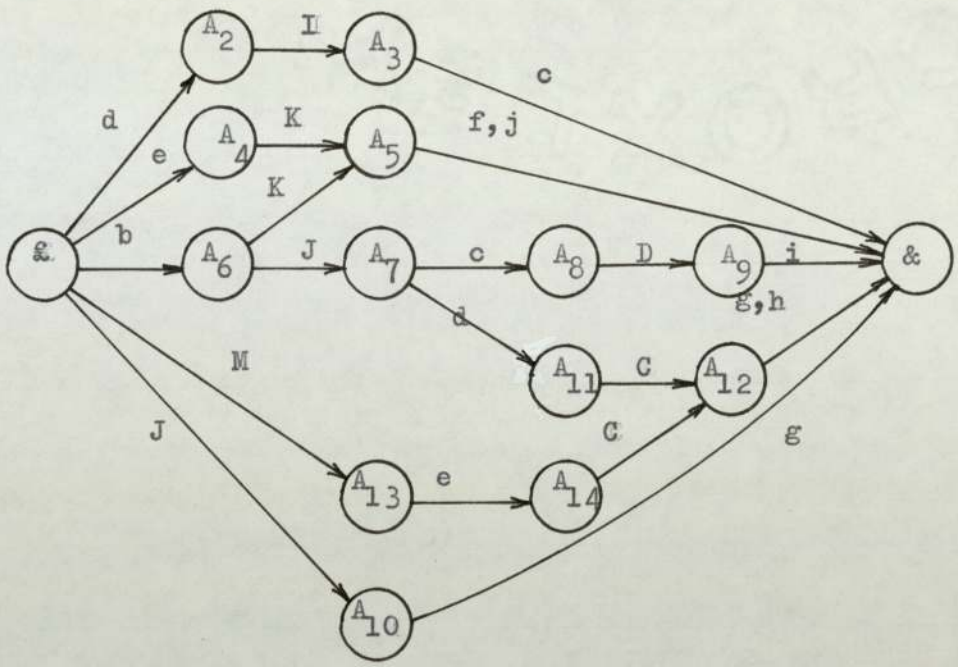


s_6=MeCh

Strings generated by G_6 : dIc
eKf
bJcDi
Jg
bJdCg
* bJdCh
MeCh
* MeCg

* denotes strings predicted by G_6

Fig.3.11 FTN corresponding to G_6



s_7=bKj

Strings generated by G_7 : dIc
eKf
* eKj
bJcDi
Jg
bJdCg
* bJdCh
MeCh
* MeCg
bKj
* bKf

* denotes strings predicted by G_7

Fig.3.12 ETN corresponding to G_7

3.3 A recognition scheme for FSG models

This section describes a scheme for the recognition of isolated words on the basis of pre-inferred rules or representative strings. The general characteristic of the recognition scheme, called scheme A, is outlined in Fig. 3.13 . Fig. 3.14 depicts a recognition system using scheme A of the recognition method and FSG's inferred from the previous section. The explanation of the system follows.

In the learning mode, algorithm 3.1 is employed to automatically construct FSG's, one for each word in the specified vocabulary. The inference process produces rewriting rules directly from the observed sample strings in response to the words spoken.Production probabilities are also estimated (see section 3.4.2) during the learning process. In the recognition mode, each unknown string presented to the recognizer is classified or decoded using the rules obtained earlier.

As shown in Fig. 3.14, the recognition process can be considered as to consist of three main levels of operation in terms of the complexity involved. The recognition always starts at the lowest level ie. level 1. A higher level is applied only if the previous one fails to classify a string according to some criteria. One sublevel is also incorporated in level 1.

Before presenting the overview of various levels of the recognition process, it is necessary to introduce some definitions which are as follows.

Definition 3.5 Parsing or syntactic analysis is the process of constructing a derivation of a string s in a grammar G ie. it is a process of finding the syntactic structure associated with the string s. The corresponding derivation tree is called a parse or a parsing-tree . If a parse can be found in a grammar G for a string s, the word

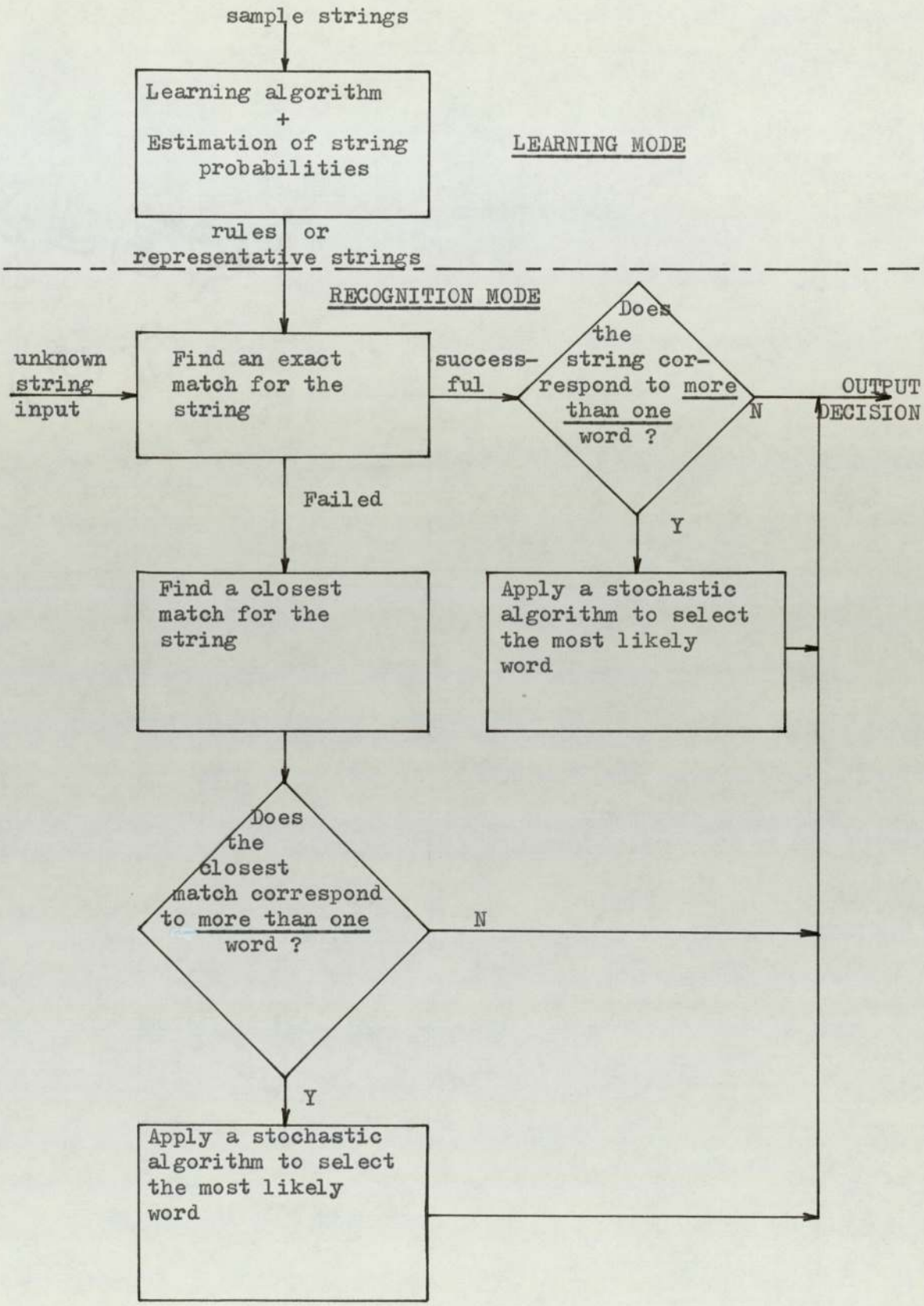


Fig. 3.13 Diagram of the recognition scheme A

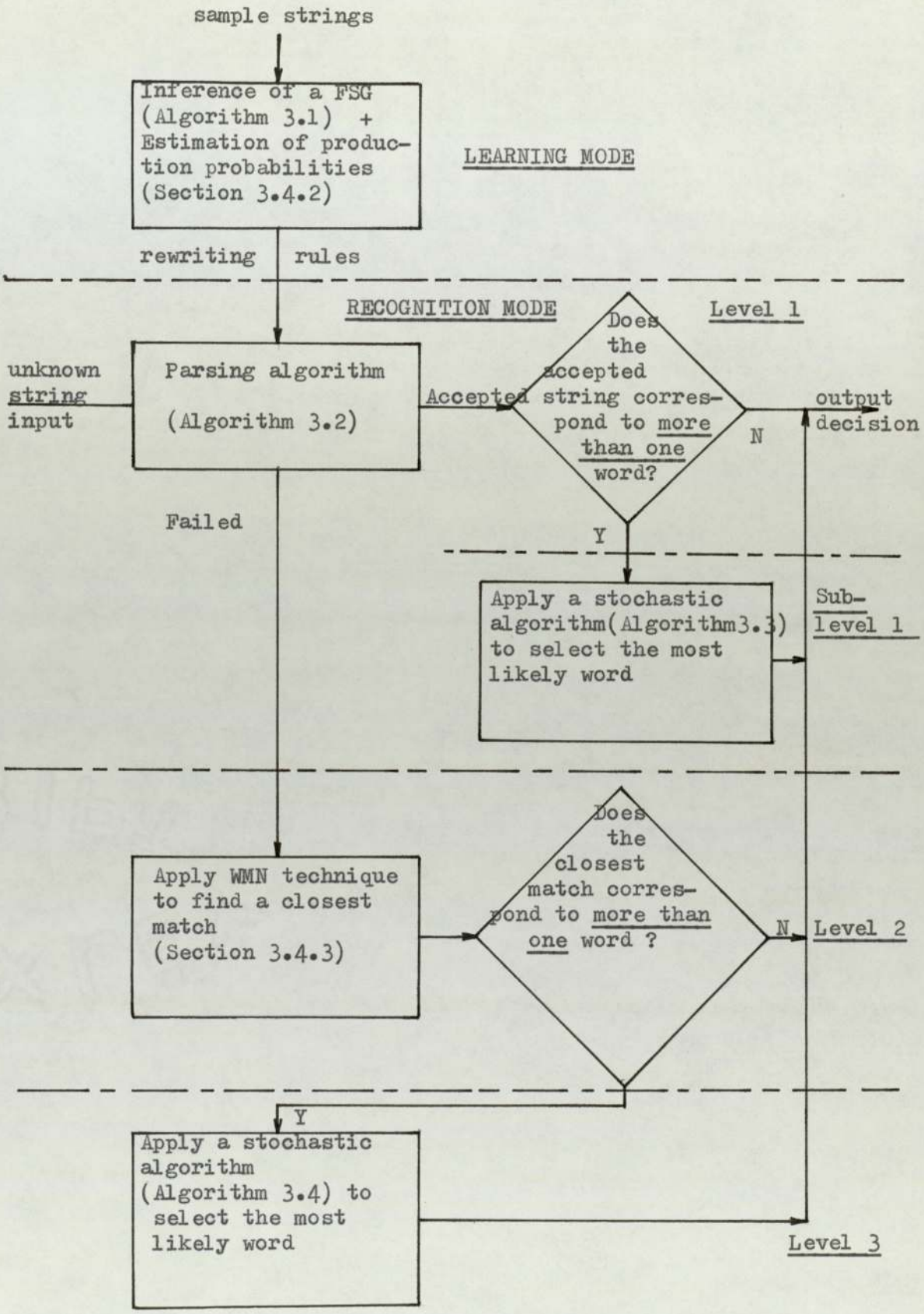


Fig. 3.14 A schematic diagram of a FSG-based recognition system using recognition scheme A

corresponding to G is said to have been recognized. Parsing can also be considered as the process of tracing through a FTN corresponding to a grammar G such that a path from the start node to the terminating node is found for a string s.

Definition 3.6 A stochastic algorithm is a finite sequence of instructions that involves the use of some statistical methods including stochastic grammars (to be defined in definition 3.7). It does not refer to an algorithm whose behaviour is uncertain or unpredictable.

A discussion concerning different levels of the recognition process is now given.

Level 1

In this simplest level of the recognition process, an unknown string is tested by means of the parsing algorithm (algorithm 3.2) to determine which grammar, if any, could have generated it. If the string is accepted by one grammar only, the corresponding word is indicated at the output. For unsuccessful parsing, the method of level 2 is then applied to decode the string. When two or more grammars can generate the string ie. the occurrence of the NE, it is necessary to employ the process of sub-level 1 to decide which grammar is the most likely to have produced the string. In this method, a stochastic algorithm (algorithm 3.3) is applied to find one 'best word' according to a maximum likelihood criterion (MLC). If two or more of such words are possible, the string is rejected.

Level 2

When the parsing algorithm in the preceding level of recognition process fails to find any grammar that can generate the unknown string, the method of the next higher level (ie. level 2) is called for. The technique of the 'weighted matching network' (WMN) is utilized to find the 'closest match' for the string ie. the grammar that could nearly have generated the string. It is basically a dynamic programming method

of optimizimg the similarity between two functions.

Level 3

This is the highest and the most complicated level of the recognition process since the operations in the two lower levels have to be performed in order to reach level 3 . It is applied when there exist two or more closest-matched words corresponding to the string. Another stochastic algorithm (algorithm 3.4) is employed to choose the most likely closest-matched word. As in level 1, the string is rejected if two or more of such words are found.

3.4 Finite-state grammar based decoding methods

This section presents in details the FSG based decoding methods or syntactic decoders for the classification of unknown strings as applied to the recognition of isolated words. The overall recognition process which employs various decoding methods in different levels of recognition operation has already been described in the previous section.

3.4.1 A parsing algorithm

The parsing algorithm to be presented is a simple top-down parse ie. it starts from the start symbol and ends with a string of terminals. In other words, the derivation of the parsing tree progresses from the root to the leaves. Backtrack facility is provided such that when a path is blocked during parsing, alternative configuration, if any, can be tried by retracing the last moves. A push-down stack is provided to store sequences of productions or rules encountered in parsing the string. The stack also aids in the backtracking process.

The following is a formal presentation of the parsing algorithm with the corresponding schematic diagram shown in Fig. 3.15 . Symbols employed in the algorithm follow the notation given earlier.

Algorithm 3.2

Step 1 Read a string $s = b_1 b_2 \ldots b_\ell$.

Step 2

(2a) Set $i = 1$

$V_{LS} = \pounds$.

(2b) Set $j = 0$.

Step 3 Set $j = j + 1$.

If $i = \ell$, go to step 7.

Step 4

(4a) Check production j of the form $A \longrightarrow aB$ whether 'A' and 'a' are identical to V_{LS} and b_i respectively.

(4b) If unsuccessful, increase j by one and go to step (4a).

If productions are exhausted, go to step 6.

Step 5 Set $V_{LS} =$ 'B' .

Put j on top of stack.

Set $i = i + 1$.

Go to step (2b).

Step 6 If the stack is empty or only one element remains in the stack, parsing fails; END.

Otherwise, pop up j from the stack.

Set V_{LS} = 'A' of rule j.

Set $i = i-1$.

Go to step 3.

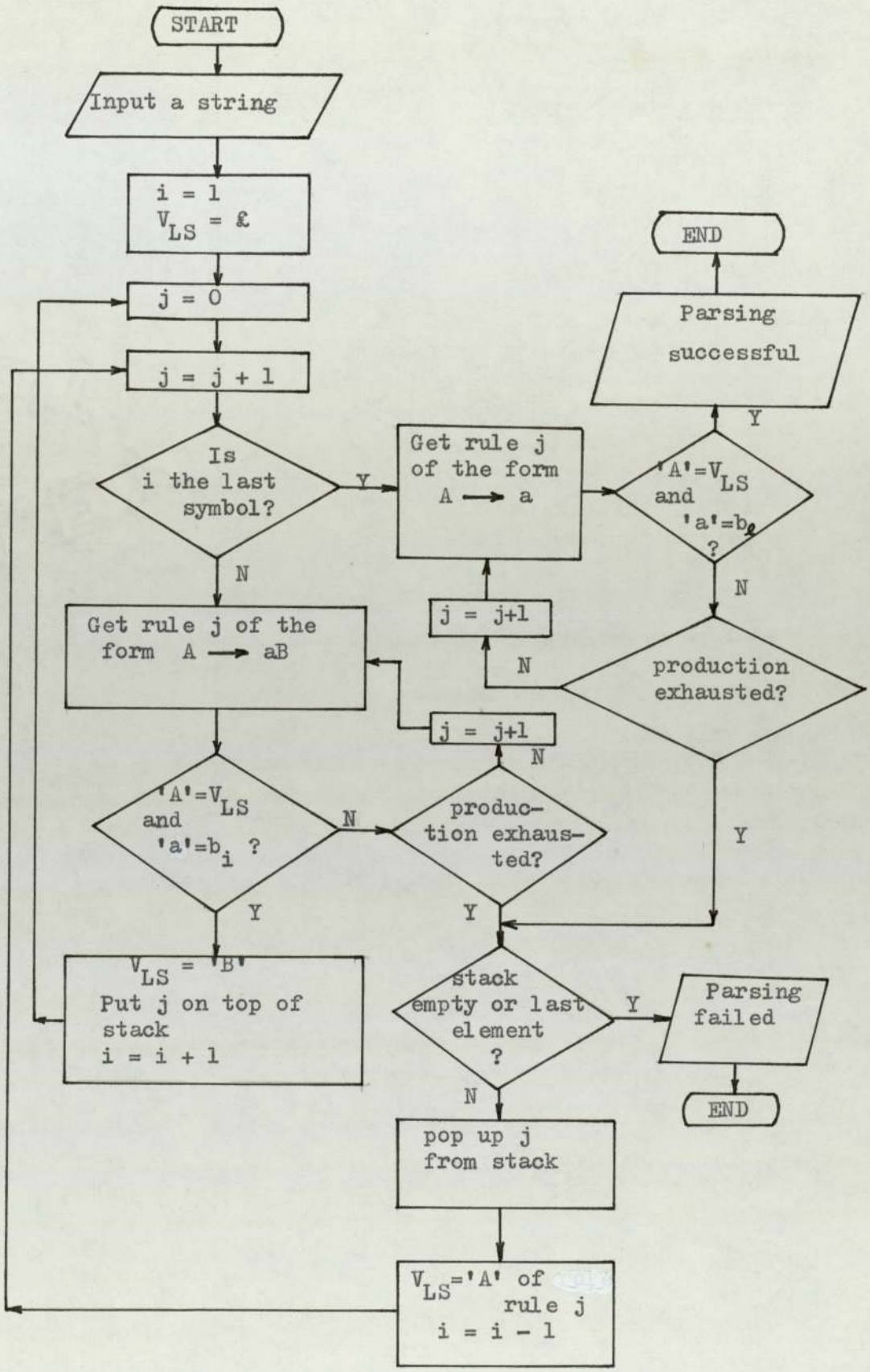


Fig. 3.15 A schematic diagram of the parsing algorithm

Step 7

(7a) Check production j of the form $A \longrightarrow a$ whether 'A' and 'a' are exactly identical to V_{LS} and b_ℓ respectively.

(7b) If unsuccessful, increase j by one and go to step (7a).
If productions are exhausted, go to step 6.
Otherwise, parsing is successful; END.

3.4.2 A maximum-likelihood criterion

The NE mentioned in the sublevel 1 of the recognition process (section 3.3) can be caused by noise or disturbances of some sort. It may also be due to the overlapping of inferred grammars which is equivalent to the overlapping of pattern classes in the case of pattern recognition. The nature of the learning algorithm and the inherent characteristics of features forming the strings are the two main causes of overlapping.

It seems that the use of phrase-structure grammars or Chomsky's grammars alone, where restrictions are placed only on the form of the productions (eg. FSG's, CFG's etc.) may not be adequate to solve the problem of the NE. Recently, there have been much research done on imposing restrictions upon the use of, in addition to restrictions on the form of the productions. The work concerning the way in which a grammar is permitted to generate strings includes, for example, an ordered grammar [69], a matrix grammar[70], a programmed grammar[71] and a grammar with a control set[72].

Another approach along this line of research is to introduce probabilities to the grammars ie. probabilistic grammars (p-grammars) [68,73,74], or stochastic grammars (s-grammars)[42,75,76]. In this

approach, probabilities are assigned to each production and each derivation of a string is associated with a probability. A more general case is a weighted grammar[76] in which some arbitrary values replace probabilities in a p-grammar or an s-grammar. Both the probabilities and the weights are usually assumed to be rational. By including probabilities into a grammar, not only the structures of different sequences of strings, but also their importance can be determined.

Formally, a stochastic finite-state grammar can be defined as follows.

Definition 3.7 A stochastic finite-state grammar (SFSG), G_s is defined as :-

$$G_s = (V_N, \Sigma, R_s, \pounds) \tag{3.1}$$

where V_N, Σ, and $\pounds$ have the same meanings as before

R_s is a finite set of stochastic productions each of the form

$$\left.\begin{array}{ll} A_i \xrightarrow{p_{ij}} aA_j & \\ \text{or} \quad A_k \xrightarrow{p_{kj}} a & \end{array}\right\} \quad \begin{array}{l} A_i, A_j, A_k \in V_N \\ a \in \Sigma \end{array}$$

where p_{ij}, p_{kj} are the production probabilities with the following properties :-

$$\sum_j p_{ij} = 1 \tag{3.2}$$

$$\sum_j p_{kj} = 1 \tag{3.3}$$

Note that a SFSG is obtained by assigning probabilities to all the rules in a given FSG. The corresponding FSG (with no probabilities attached to the rules) is called a characteristic grammar .

Since the only input data available to the recognizer is the positive information sample set, the required production probabilities of a SFSG have to be estimated from this sample set. Techniques for the determination of rule probabilities for unambiguous grammars have been developed based on a maximum likelihood estimation[63,66,77].

The first stage of this method involves the parsing of each and every string in the sample set. For each production ij (ie. a link from node i to node j in the corresponding FTN), a count is made of the number of times that rule ij is used in the derivation of all strings in the sample set. The production probability of link ij is then obtained from the ratio of the above count to the number of times that all the links originating from node i are used in parsing the same set of strings.

More formally, the technique for the estimation of production probabilities can be described as follows :-

1. Let the sample strings (all distinct) be

$$S_+ = (s_j \mid j = 1,2,\ldots,M_D) \tag{3.4}$$

where M_D is the number of distinct strings in set S_+.

2. f_j, the estimated probability of string s_j is determined by the relative frequency of its occurrence ie.

$$f_j = F_j / M_S \tag{3.5}$$

where F_j = number of string s_j

M_S = number of total strings.

3. For a production $A_i \longrightarrow \alpha_k$ in grammar $G = (V_N, \Sigma, R, \pounds)$ where $\alpha_k = aA_k$ or a ; find $N_{ik}(s_j)$, the number of times that production $A_i \longrightarrow \alpha_k$ is used in parsing string s_j.

4. n_{ik}, the expected number of times that rule $A_i \longrightarrow \alpha_k$ is used in parsing <u>all</u> the sample strings in S_+ is given by :-

$$n_{ik} = \sum_{j=1}^{M_D} f_j N_{ik}(s_j) \tag{3.6}$$

5. The maximum-likelihood estimate for p_{ik}, the production probability of rule $A_i \longrightarrow \alpha_k$ is obtained by

$$p_{ik} = n_{ik} \Big/ \sum_k n_{ik} \tag{3.7}$$

A formal proof of the above result given by equation 3.7 can be found in, for example, references 42 and 68.

The estimated production probabilities thus obtained are added to the rules of the characteristic grammar G to form the required SFSG. The method just described is expected to be adequately accurate when applied to the problem of the approximations of an ill-defined FE by means of formal grammars. In the actual implementation, some saving of the execution time can be achieved by incorporating the counting operations into the learning process. This follows because the parsing of strings is done at the same time as the characteristic grammar is being inferred.

The following example illustrates the estimation process.

Example 3.3 Let the strings in the sample set $S_{3.3}$ be :-

s_1 = Lg	s_2 = Lh	s_3 = Lg
s_4 = Lg	s_5 = KcCd	s_6 = Lh
s_7 = Jf	s_8 = Nh	s_9 = Mℓ
s_{10} = Kd	s_{11} = JcDe	s_{12} = Nℓ .

By applying algorithm 3.1, the following characteristic grammar is constructed.

$G_{3.3} = (V_N, \Sigma, R, £)$

where $V_N = (£, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9)$

$\Sigma = (C, D, J, K, L, M, N, c, d, e, g, h, j, ℓ)$

R = all the rules in table 3.1

The expected number of times that each rule in R is used in the derivation of all of the above strings together with the corresponding estimated production probability are given in table 3.1 .

Rule ij	n_{ij}	p_{ij}
$\pounds \rightarrow LA_2$	5	5/12
$\pounds \rightarrow KA_3$	2	2/12
$\pounds \rightarrow JA_6$	2	2/12
$\pounds \rightarrow NA_7$	2	2/12
$\pounds \rightarrow MA_8$	1	1/12
$A_2 \rightarrow g$	3	3/5
$A_2 \rightarrow h$	2	2/5
$A_3 \rightarrow cA_4$	1	1/2
$A_3 \rightarrow d$	1	1/2
$A_4 \rightarrow CA_5$	1	1/2
$A_4 \rightarrow DA_9$	1	1/2
$A_5 \rightarrow d$	1	1
$A_6 \rightarrow j$	1	1/2
$A_6 \rightarrow cA_4$	1	1/2
$A_7 \rightarrow h$	1	1/2
$A_7 \rightarrow \ell$	1	1/2
$A_8 \rightarrow \ell$	1	1
$A_9 \rightarrow e$	1	1

Table 3.1 The estimated production probabilities of rules in example 3.3

The problem of the NE where a string s_j corresponds to two or more words can be handled as follows.

Applying a maximum likelihood criterion, a given string s_j is classified as corresponding to word w_k if and only if $P(w_k/s_j)$ is a maximum.

From Bayes' rule[78], $P(w_k/s_j)$ can be expanded into :-

$$P(w_k/s_j) = P(w_k,s_j).P(s_j) \qquad (3.8)$$

Since $P(s_j)$ is constant for a given s_j, only the term $P(w_k,s_j)$ needs to be maximized.

Rewriting $P(w_k,s_j)$ using Bayes' rule yields :-

$$P(w_k,s_j) = P(s_j/w_k) \cdot P(w_k) \qquad (3.9)$$

To ensure that there exists no initial bias towards any particular word, the a priori probability $P(w_k)$ is assumed to be equal for every word in the vocabulary.

Thus only the term $P(s_j/w_k)$ is required to be maximized if and only if s_j is to be classified to word w_k, where $P(s_j/w_k)$ is the product of all production probabilities correspond to word w_k used in parsing s_j.

Summarizing : In the case of the NE, a given string is classified to the word with the 'maximum-likelihood' probability obtained from the product of all production probabilities of the corresponding grammar which are used in the derivation of that string.

The following presents an algorithm for dealing with the NE based on a MLC previously described. The corresponding schematic diagram is shown in Fig. 3.16 .

Algorithm 3.3

Step 1 Read $(w_i \mid i=1,\ldots,W_N)$, all the words that correspond to the string s_j ;

where W_N is the number of such words.

Set $k = 0$.

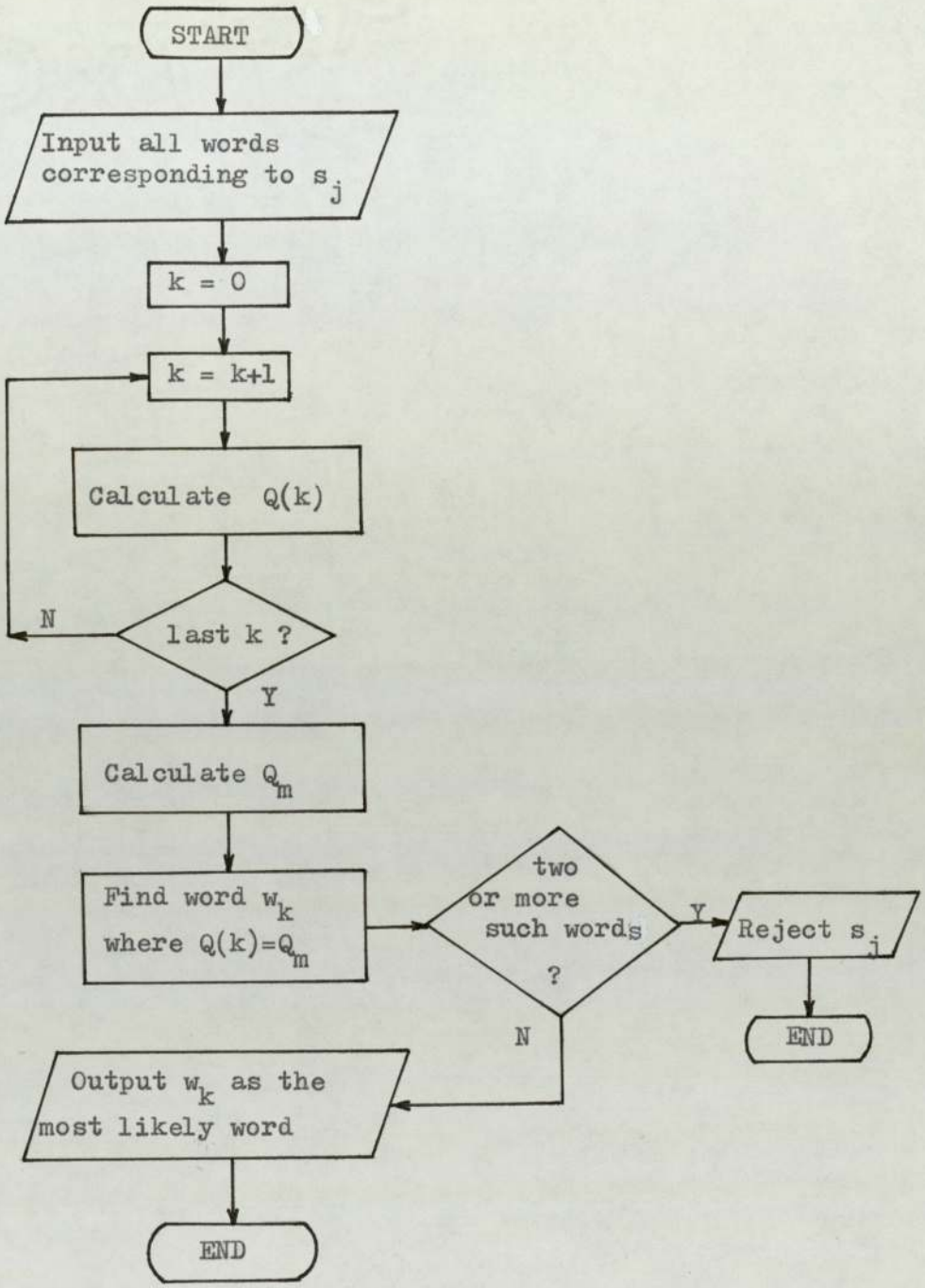


Fig. 3.16 A schematic diagram of algorithm 3.3

Step 2 Set $k = k + 1$.

Calculate and store $Q(k)$

where $Q(k) = \prod_{i=1}^{I(s_j)} p_i(s_j)$

where $I(s_j)$ = number of steps in the derivation of s_j

$p_i(s_j)$ = probability of the production used at the ith step of the derivation of s_j .

Step 3 If $k = W_N$, go to step 4.

Otherwise, go to step 2.

Step 4 Calculate $Q_m = \max_{k=1}^{W_N} Q(k)$.

Step 5 If there are two or more words associated with Q_m, reject s_j; END.

Otherwise, decide that s_j corresponds to word w_k if $Q(k) = Q_m$; END.

3.4.3 A 'weighted matching network' technique

The unsuccessful decoding of a string by the parsing algorithm (algorithm 3.2) may be caused by one or more of the following factors. The word spoken (and hence the string representing that word) may not be in the vocabulary. If the word is known to be outside the vocabulary and it is intended to add the new word to the existing vocabulary, then a new grammar has to be created to accomodate that word. It is also possible for errors to appear in the string. This may be due to noise or some disturbance or free variation of speech as a result of the speaker's characteristics. In addition, the ambiguity of speech signal and procedures of segmentation and labelling may also induce errors. In the case of telephone-grade speech, the string is subjected to an

even greater chance of being corrupted. This is caused by various characteristics of the telephone system such as the restricted bandwidth, background noise and impulsive noise, nonlinear distortion, variation of sensitivity and gain and so on.

From the foregoing discussion, it appears that the simple top-down parsing technique is inadequate for dealing with errors. This is because it can only indicate the presence of errors but not their locations. This section presents a WMN technique for finding a closest match between the corrupted string and the strings derivable from grammars with a facility for pinpointing errors. The technique is well suited for applying to FSL's since it is based on the concept of FTN.

Before presenting the WMN and its associated technique, formal definitions are now given of different types of symbol errors or symbol alterations.

Definition 3.8 A deletion error is one which causes the correct input symbol a_j to appear as λ at the output, where λ is the null string symbol. In other words, a_j is deleted from the input string.

Definition 3.9 An insertion error is one which causes an extra symbol b_k to be inserted into the current string.

Definition 3.10 A substitution error is one where the correct input symbol a_j is replaced by a symbol b_k which appears at the output.

Concept of a distance concerning the above types of errors is defined next.

Definition 3.11 The minimum number of symbol alterations consisting of any combination of deletion, insertion and substitution errors, needed to convert an observed string x to a prototype string y is known as the Levenshtein distance (LD) [79].

If various weights are assigned to each of the symbol

alterations, the corresponding distance becomes weighted Levenshtein distance (WLD) .

The WMN technique is basically a method of optimizing the similarity between two functions. The first function is an observed string whilst the second is the set of all strings that can be generated by the grammar concerned ie. the dictionary. The aim is therefore to determine the LD or the WLD of an observed string and a given grammar.

The problem of spelling correction by matching a given string with the dictionary has been studied by many researchers including (34,35,80-84,88). The common approach to solving the problem as given by Velichko and Zagoruyko(81) is based on the construction of a 2-D array. The array is formed by associating one function with one axis or one dimension of the array and the other function with another axis or dimension. The principle of dynamic programming(85) is then applied to search through the array for an optimal solution.

Descriptions are now made concerning the WMN and how it can be used to obtain the required closest match (ie. LD or WLD).

The WMN, also a 2-D array of the kind mentioned above, can be constructed as follows. First, a FTN is built from the FSG under consideration. This forms the first row of the WMN. The remaining rows are then obtained by repeating the FTN ℓ times directly below the first row, where ℓ is the length of an observed string. The overall structure just created becomes an array of $(\ell + 1) * m$ nodes where there are m nodes in the FTN.

The next stage involves the connections between nodes in adjacent rows which is accomplished in the following manners.

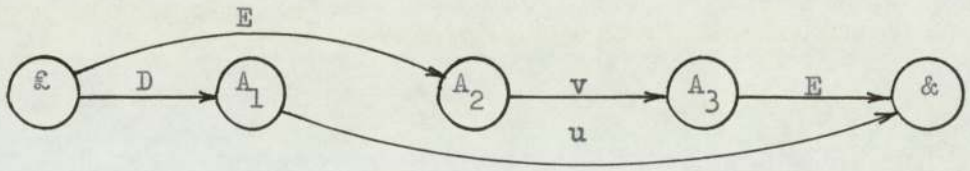
(i) There is a link from each and every node in row k to each and every node directly underneath in row k+1 .

(ii) For every transition ij from node i to node j in the FTN, there is a link from node i in row k to node j in row k+1.

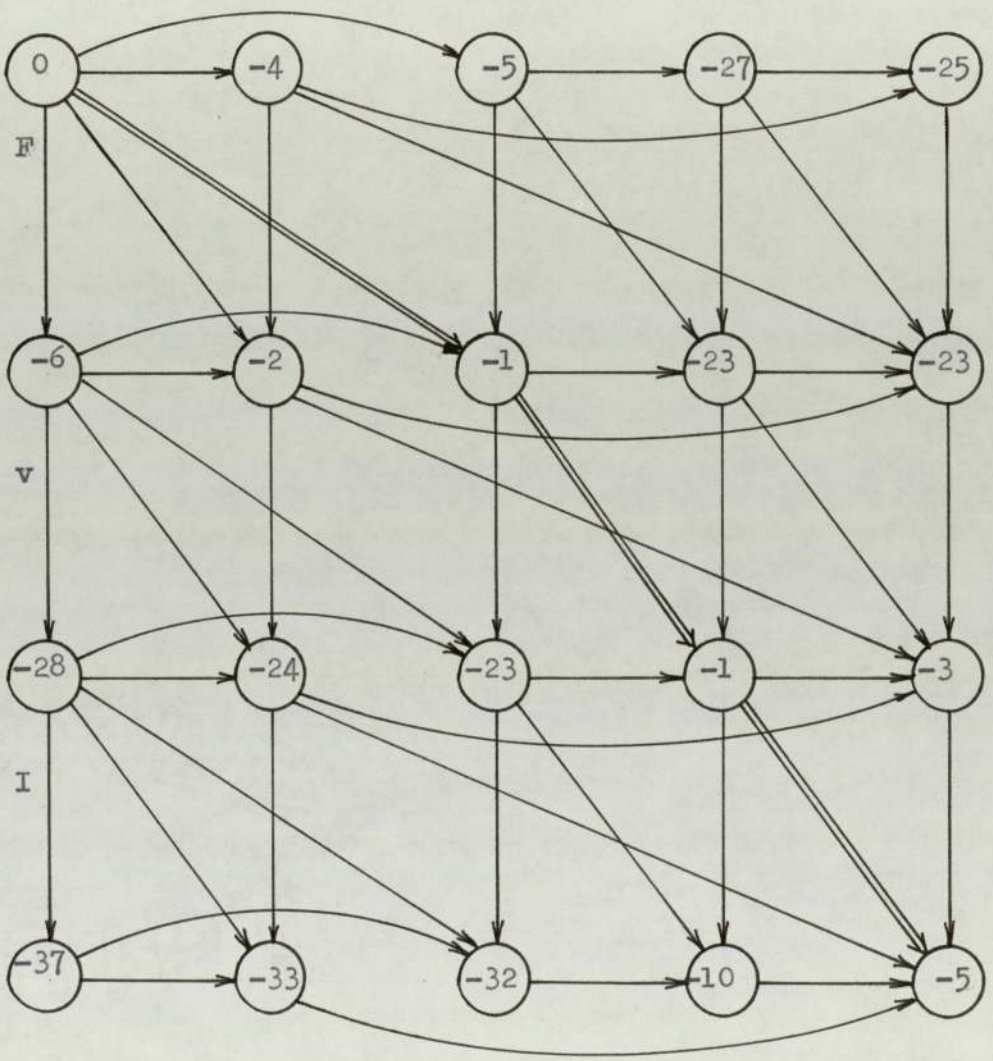
The above procedures are applied recursively starting from k = 1 to k = ℓ.

This completes the construction of the WMN except for the determination of the content of each element in the array which will be dealt with later. Fig. 3.17 exemplifies the WMN for a length-3 string and a given FTN. Appendix B gives numerical values of significance of various symbols. As illustrated by double-lined arrows, an optimal path always starts from the top left node (ie. element (1,1) of the array) and terminates at the bottom right node (ie. element (ℓ+1,m)). Thus, the general direction of a path in the WMN is from top to bottom and left to right. The number inside each node represents the minimum penalty incurred in traversing from node (1,1) to that node. Consequently, the content of node (1,1) is always zero. The minus sign indicates that the number concerned is a penalty and not a reward. The absolute value of element (ℓ+1,m) gives the LD or WLD as required (it is WLD in Fig. 3.17).

Various types of symbol errors defined earlier can be graphically represented by the WMN in the following ways. A horizontal link denotes an omission of a symbol associated with that link from an observed string ie. it represents a deletion error. A symbol extraneously inserted into an observed string resulting in an insertion error is depicted by a vertical link. For a diagonal link ij connecting node i in row k and node j in row k+1, a substitution error occurs if and only if the kth symbol in the observed string is not identical to the symbol corresponding to link ij in the FTN.



(a) a FTN corresponding to a FSG



(b) a WMN obtained from FTN in (a)

The required optimal path is designated by double-lined arrows.

Fig. 3.17 Matching of a string with a FSG using WMN technique

A method is now presented for computing the contents of various nodes in the WMN in the search for an optimal path.

The process is divided into two stages. The first stage is concerned with the calculations of the first row of the WMN whilst the second stage determines the contents of elements in subsequent rows.

Stage I Computation of elements in the first row

Let $e(i,j)$ be the element in row i and column j of the WMN.

Given that $e(1,1) = 0$.

All other elements in the first row of the WMN are calculated iteratively using the following equation :-

$$e(1,j) = \underset{\substack{\text{all i in} \\ A_i \longrightarrow a_j A_j \\ A_i \longrightarrow a_j}}{\text{Max}} \left[e(1,i) - F_D(j) \right] \qquad (3.10)$$

where

indices i and j denote the column positions corresponding to A_i and A_j respectively in production $A_i \longrightarrow a_j A_j$.

$j = m$ for production $A_i \longrightarrow a_j$.

$F_D(j)$ is a non-negative 'deletion function' defined as :-

$$F_{D(j)} = \left| \text{significance of '}a_j\text{'} \right| \qquad (3.11)$$

where significance of 'a_j' is the weight associated with 'a_j' obtained from a priori knowledge of the symbol 'a_j' .

Stage II Computation of elements in row k $(1 < k \leq \ell + 1)$

(i) Due to the previous row ie. row k-1

$$e^{(i)}(k,j) = \text{Max} \left\{ \underset{\substack{\text{all i in} \\ A_i \longrightarrow a_j A_j \\ A_i \longrightarrow a_j}}{\text{Max}} \left[e(k-1,i) - F_S(jk) \right] , \; e(k-1,j) - F_I(k) \right\} \qquad (3.12)$$

where $F_I(k)$ is a non-negative 'insertion function' given by :-

$$F_I(k) = \left|\text{significance of } b_{k-1}\right| \qquad (3.13)$$

where b_{k-1} is the (k-1)th symbol in the observed string.

$F_S(jk)$ is a non-negative 'substitution function' defined as:-

$$F_S(jk) = \left|(\text{significance of } b_{k-1}) - (\text{significance of } 'a_j')\right| \qquad (3.14)$$

(ii) Due to the same row ie. row k

$$e^{(ii)}(k,j) = \text{Max}\left\{ e^{(i)}(k,j), \underset{\substack{\text{all i in} \\ A_i \longrightarrow a_j A_j \\ A_i \longrightarrow a_j}}{\text{Max}} \left[e^{(i)}(k,i) - F_D(j) \right] \right\} \qquad (3.15)$$

The procedure in stage II is repeated until the last row (ie. $k = \ell + 1$) has been processed.

The final value of element $(\ell + 1, m)$ obtained from the above computations indicates the minimum penalty incurred if the observed string is to be generated from a given set of rewriting rules. This is the case of finding WLD whose value together with the contents of other elements of WMN are shown in Fig. 3.17 .

To obtain the LD, substitute the followings in equations 3.10, 3.12 and 3.15 :-

$$\left.\begin{aligned} F_D(j) &= 1 \\ F_I(k) &= 1 \\ F_S(jk) &= \begin{cases} 1 & \text{if and only if } b_{k-1} \neq a_j \\ 0 & \text{otherwise} \end{cases} \end{aligned}\right\} \qquad (3.16)$$

The technique of the WMN can be applied to select one best word which is the closest match to an observed string as follows. Find WLD for every word in the vocabulary and choose the word with the smallest value of WLD as the required best word.

In concluding this section, some issues concerning the WMN are discussed below.

(a) It is possible to obtain two or more optimal paths for an observed string and a given set of rules. This implies that the string selected from the grammar to represent the observed string as the prototype string may or may not correspond to an original error-free string. However, this poses no problem to the application in the recognition of isolated words. Provided representative strings are generated by the same set of rules, the word selected as a best word is always the same irrespective of which representative string is chosen.

(b) The method of WMN can, of course, be employed to test whether an observed string is derivable from a FSG. That is, it can be used to perform the function of the parsing algorithm (algorithm 3.2) . The string is said to have been generated by the grammar concerned if, and only if, the element $(\ell + 1, m)$ of the WMN is zero. In general, the time required to parse a string is longer for the WMN technique than that required for the parsing algorithm. Therefore, in applications where it is required to know only the existence of errors but not their locations, the simple parsing method is preferable to the method of WMN.

3.4.4 A stochastic algorithm

As pointed out in section 3.3, the WMN technique is applied to decode an observed string which has been unsuccessfully analysed by the parsing algorithm. Even in this case, it is still possible for two or more words to be assigned the same value of WLD. An example of this type of situation is illustrated in Fig. 3.18 . As before,

significance of each symbol can be determined from appendix B.

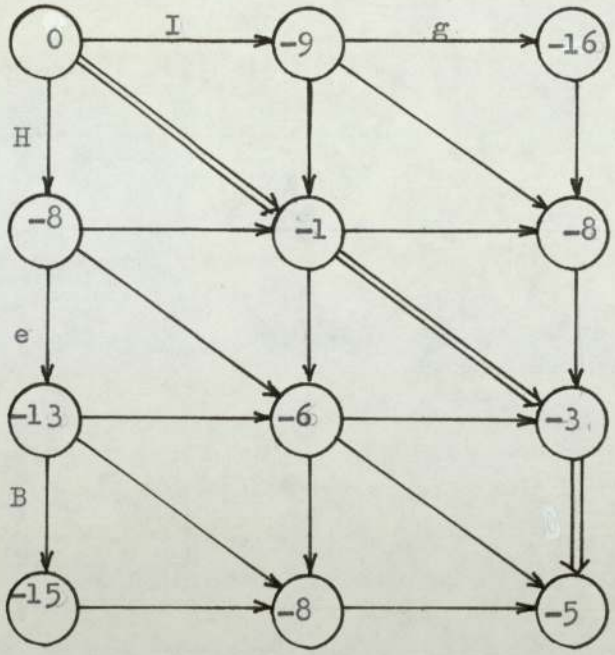
This section presents a decision criterion using probabilities for the selection of a word that is most likely to represent the string.

From the previous section, the overall penalty of an optimal path is obtained by adding the contributions from each individual link comprising that path. Likewise, the method to be presented below is based on the same assumption, namely, that an optimal path can be divided into independent links. All of the individually optimized links are then combined to form a final optimal solution.

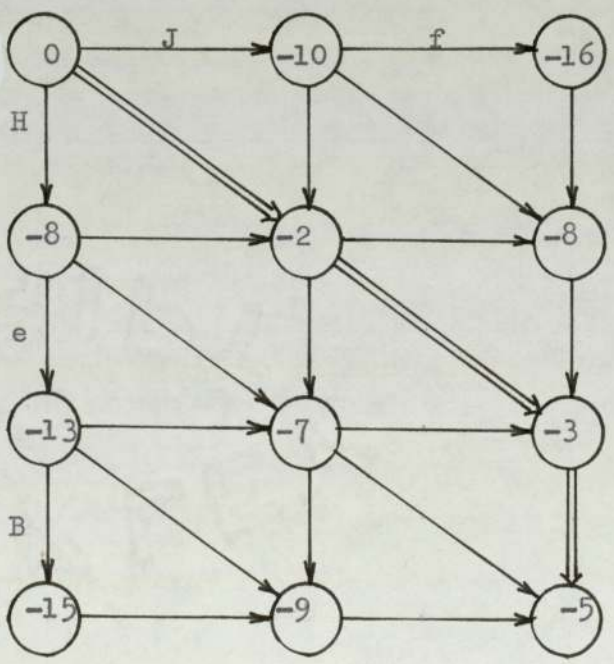
In outline, the method works as follows.
For each link in an optimal path of the WMN which contributes an error, an estimation is made of the likelihood of that symbol alteration. The final result of an optimal path is the product of probabilities estimated above of all such links comprising that path. The procedure is repeated for all optimal paths appearing in the WMN. Results of each path are then combined to give the final solution.

The above process is applied to all eligible words ie. words associated with the same WLD. The word with the largest value of the final solution is then selected as the one most likely to generate the string. If two or more such words are possible, the string is rejected as before.

Techniques for the estimation of probabilities associating with each type of symbol alterations are given next.



(a) a WMN of word 1



(b) a WMN of word 2

Fig. 3.18 An example of two words sharing the same value of WLD

(a) Deletion errors

Let $P_D(a_{ij})_k$ be the probability that a_{ij}, the terminal symbol corresponding to link ij is deleted from a string in word w_k. This is subject to the followings :-

(i) deletion coefficient t_D which determines how probable a particular symbol alteration is due to deletion event (rather than substitution or insertion).

(ii) the conditional probability $p(a_{ij}/w_k)$, the probability that the symbol deleted is a_{ij} given that it is in a string corresponding to word w_k.

The selection of $p(a_{ij}/w_k)$ rather than $p(w_k/a_{ij})$ results from the rewriting of $p(w_k/a_{ij})$ using derivations similar to equations 3.8 and 3.9 and the application of the assumption given in section 3.4.2 .

The criterion for the case of deletion errors is as follows :-
Decide that the symbol deleted is a_{ij} corresponding to word w_k if $P_D(a_{ij})_k$ is the largest for all words, where

$$P_D(a_{ij})_k = t_D \cdot p(a_{ij}/w_k) \qquad (3.17)$$

(b) Insertion errors

Let $P_I(b)_k$ be the probability that the terminal symbol b is inserted into an observed string in word w_k. This is governed by :-

(i) insertion coefficient t_I .

(ii) the frequency of occurrence of the inserted symbol b, p(b). That is, the more frequent b occurs, the more likely that it is effected by noise or disturbance etc.

The conditional probability $p(b/w_k)$ is not used in this case. This is because, it has been determined experimentally that any symbol can be inserted into a string corresponding to any word,

even though that particular symbol may never appear in that word during the training mode.

The criterion for the case of insertion errors is :-
Decide that the string, where a symbol b has been inserted, corresponds to word w_k if $P_I(b)_k$ is the largest for all words,
where

$$P_I(b)_k = t_I \cdot p(b) \qquad (3.18)$$

(c) Substitution errors

Let $P_S(a_{ij})_k$ be the probability that a_{ij}, the terminal symbol associated with link ij is substituted by a symbol b .
This is affected by :-

(i) substitution coefficient t_S .

(ii) the conditional probability $p(a_{ij}/w_k)$.

The probability of a symbol b is not considered since the importance of b relative to a_{ij} has already been taken into account in the WMN.

Thus, the criterion for the case of substitution errors is :-
Decide that the symbol being substituted (by a symbol b) is a_{ij} corresponding to word w_k if $P_S(a_{ij})_k$ is the largest for all words,
where

$$P_S(a_{ij})_k = t_S \cdot p(a_{ij}/w_k) \qquad (3.19)$$

The following is an algorithm for finding the most probable word, whose corresponding grammar could nearly have generated an observed string. The techniques described above are employed to estimate the required probabilities of various symbol alterations. The associated flow diagram is depicted in Fig. 3.19 .

Algorithm 3.4

Step 1 Read $(w_i \mid i=1,\ldots,W_N)$, all the words corresponding to string s_j with the same smallest value of WLD.

Set $k = 0$

$K = 0$, where K is the optimal path number .

Step 2

(2a) Set $k = k + 1$.

(2b) Set $K = K + 1$.

For each symbol alteration found in path K and word w_k, calculate $P_D(a_{ij})_k$, $P_I(b)_k$ or $P_S(a_{ij})_k$ depending on the type of the error.

Step 3 Calculate P_K^k which is the product of any combination of $P_D(a_{ij})_k$, $P_I(b)_k$ and $P_S(a_{ij})_k$ estimated from path K and word w_k.

Step 4 If current value of K is the last one, go to step 5.

Otherwise, go to step (2b).

Step 5 Calculate $P_T^k = \underset{K}{\leq} P_K^k$

Step 6 If $k = W_N$, go to step 7.

Otherwise, go to step (2a).

Step 7 If $\underset{k}{\text{Max}}\ P_T^k$ corresponds to two or more words, reject s_j; END.

Otherwise, decide that s_j corresponds to word w_k if P_T^k is the largest for all the words ; END.

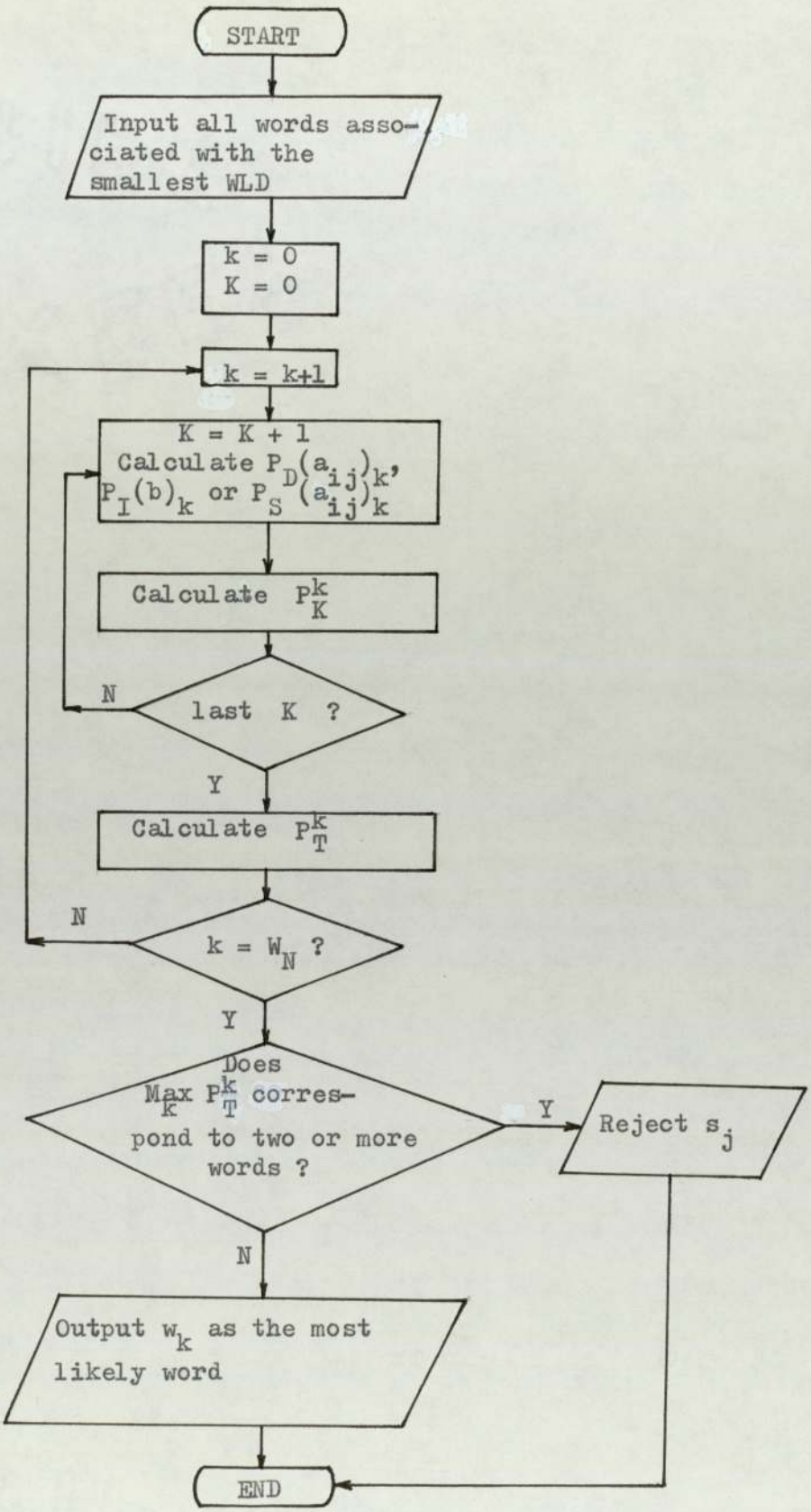


Fig. 3.19 A schematic diagram of algorithm 3.4

CHAPTER 4

CONTEXT-FREE GRAMMAR-BASED MODELLING

4.1 Motivation

The first application of CFG's to programming languages was probably made by Backus[86] in specifying the syntax of the ALGOL language. Since then, the use of CFG's has become common among research workers in the computing field. In contrast, as already mentioned in section 2.2, it is not clear at present what type of grammar best represents strings of symbols associated with an IWR system. In principle, the use of a nonrecursive FSG would be adequate to model the FE where only finite-length strings are involved. However, it is possible that the use of less-restricted grammars may provide models which are preferable in some way. For example, a model constructed on the basis of a CFG may have fewer nodes and/or links than a FSG-based model using the same data.

The above is analogous to digital filtering. It is possible, in principle, to use a nonrecursive digital filter whenever a finite-length impulse response is required. However, it is sometimes better to use a recursive filter despite the type of the required impulse response. This is because the volume of necessary computation may be greatly reduced[87].

In addition, the use of a CFG provides the model with a push-down mechanism which makes it possible to temporarily suspend the processing of a constituent of a language or a string at a given level. This is done so that an embedded constituent can be processed using the same grammar. The foregoing operation thus allows many regularities of the language, if any, to be captured. As an example, a substring occurring in a number of different contexts may be represented by a single

rule instead of by a number of independent rules for each of the different contexts. For the above reasons, it is of interest to have methods for the construction of nonrecursive CFG's.

This chapter is concerned with the application of CFG's to the modelling of a FE in the recognition of isolated words. The general approach of the CFG inference problem follows the outline of the approach based on FSG's as given in Fig. 3.1 . The FSG depicted in Fig. 3.1 is, of course, replaced by a suitable CFG. A method is given for the direct construction of a proper nonrecursive CFG from a set of sample strings. The inference method to be presented generates compact CFG's having a near minimal number of rules and/or nonterminals, compatible with the requirement to be able to generate all strings in the sample set.

The basis of the inference method involves a comparison between an incoming string and an existing CFG. The matching process requires the computation of the minimisation matrix, M, (to be defined later) whose elements reveal the compatibility or otherwise between the string, its substrings and the grammar. If any incompatibility exists, appropriate rules and/or nonterminals and terminals are appended such that the augmented CFG can generate the string.

In contrast to enumerative techniques, the method mentioned above is computationally efficient. This is because it is based on direct construction of a grammar from sample strings. In addition, the method is conformed to basic assumptions about an inference process in general as given in section 2.2 . Suitable decoding techniques are also described.

4.2 Graphical representation of a CFG

In this section, a method is given for the construction of a network, which is essentially a set of FTN's applied recursively, to represent a CFG in a graphical format. The network is fully explained in the following definition.

Definition 4.1 A push-down transition network (PTN) is a collection of directed graphs with a unique start node and a special set of terminating nodes together with a push-down store. The network consists of a principal graph which contains the start node and optionally, a number of auxiliary graphs . Each and every link or transition in the network is associated with either a terminal or a nonterminal but not both.

Transitions involving nonterminals and a push-down stack can be interpreted as follows.

If a nonterminal C is encountered during a transition from node A to node B, the processing of an observed string at the present level is temporarily suspended. This is followed by the saving of the nonterminal associated with node B on a push-down stack. The processing then resumes with the new transition, commencing from the state or node corresponding to nonterminal C which is either in the present graph or in another graph. In other words, the transfer of control from one level of the process to another can be viewed as a procedure of a subroutine call to another graph or the current one. Upon reaching a terminating node, the symbol on top of the stack is poped-up, removed from the stack and used as the new starting point. An attempt to pop up an empty stack after the last symbol of a string has just been processed signifies the acceptance of the string.

All the CFG's considered in this thesis are assumed to be proper so as to eliminate as many unnecessary rules as possible. The formal definition follows.

Definition 4.2 A CFG $G = (V_N, \Sigma, R, \&)$ is said to be proper[55] if :

(a) R has no λ-productions ($A \longrightarrow \lambda$ for all A in V_N) ie. G is λ-free,

(b) there is no derivation of the form $A \overset{+}{\Longrightarrow} A$ where $A \in V_N$ ie. G is cycle-free, and

(c) G has no useless symbols ie. there does not exist a nonterminal that does not generate any terminal strings.

A proper CFG can be transformed to a PTN in the following manners.

(i) Partition the rules of a given CFG into groups of rules where rules in each group have identical LS nonterminals.

(ii) Construct the principal graph, beginning with the set of start rules, based on the constraints given below.

(iii) Apply the same constraints to the remaining sets of rules, if any, to obtain appropriate auxiliary graphs.

The general procedures employed in the construction of a PTN follow those of a FTN. The following describes constraints governing the creation of links and/or nodes of a PTN from a set of rules of a CFG.

A rule in the CFG can be in one of the following forms : -

Form 1 $A \longrightarrow \alpha B$ where $\alpha \in V^+$; A, B $\in V_N$. (4.1)

Form 2 $A \longrightarrow \beta a$ where $\beta \in V^*$ and $a \in \Sigma$. (4.2)

For a given rule, there are as many links as the number of elements in α or β ; except when β is a null string where there will be exactly one link. Each link created from and associated with an

element of α or β connects the current node to a new node except for the last element of α. In that case, the link corresponding to the last element of α terminates at the node associated with nonterminal 'B'.

For a rule of the second form, the final link of the rule, associated with a terminal 'a', ends at one of the terminating nodes.

The above procedure can be illustrated by the following example.

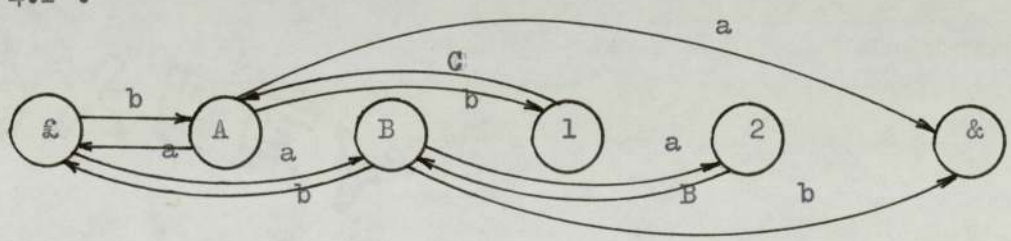
Example 4.1 Consider a CFG $G_{4.1} = (V_N, \Sigma, R, £)$ whose rules have already been partitioned into various groups as given below.

R :				
	£ ⟶ bA	A ⟶ bCA	B ⟶ aBB	C ⟶ cB
	£ ⟶ aB	A ⟶ a£	B ⟶ b£	C ⟶ c
		A ⟶ a	B ⟶ b	

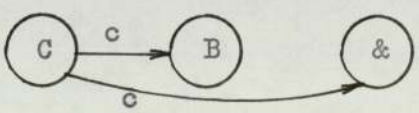
where $V_N = (£, A, B, C)$

$\Sigma = (a, b, c)$

The PTN constructed using the above procedure is depicted in Fig. 4.1 .



(a) the principal graph



(b) an auxiliary graph

Fig. 4.1 A PTN corresponding to the CFG in example 4.1

UNIVERSITY
LIBRARY

4.3 Computation of the minimisation matrix

It is usually more difficult to deal with the problem of grammar inference of a CFG than that of a FSG. This is because many properties that are decidable for a FSG become undecidable for a CFG. In particular, it is known [54] that a general algorithm does not exist to test whether two CFG's are equivalent. For these reasons, many CFG inference algorithms are confined to specific types of CFG's. Likewise, the CFG's to be inferred are assumed to be in a normal form whose definition follows.

Definition 4.3 A CFG in Chomsky normal form[58] is one in which the productions are of the following forms only.

$$A \longrightarrow BC \qquad \text{where } A, B, C \in V_N \qquad (4.3)$$

$$A \longrightarrow a \qquad \text{where } a \in \Sigma \qquad (4.4)$$

Rewriting rules of the first form are called bielement rules, and those of the second form are known as terminating rules .

Any CFG can be converted into an equivalent CFG in Chomsky normal form [54,55] so that no generality is lost by dealing only with CFG's in this form. An example of such transformation is given in reference 54.

Before the inference method can be given, it is necessary to describe the minimisation matrix, M, which forms the basis of the inference process. This is done in the following two sections where iterative procedures for the computation of the nonweighted and weighted versions of the M-matrix are presented respectively.

4.3.1 Nonweighted M-matrix

This section describes the nonweighted version of the M-matrix and presents an iterative procedure for the computation of its elements. Discussion of the matrix is now given.

Definition 4.4 Let $s = b_1 b_2 \ldots b_\ell$ be a string of length ℓ, where $b_i \in \Sigma$ for $i=1,\ldots,\ell$. For a given CFG in Chomsky normal form and for a string s, the minimisation matrix (nonweighted) , M (hereafter referred to as M-matrix) is a three dimensional, $\ell * \ell * r$ matrix. r is the number of nonterminals in the grammar. Element m_{ijk} of M denotes the minimum number of symbol alterations (any combinations of deletions, insertions or substitutions) required if the length-i substring of s, whose first symbol is b_j, is to be generated by the grammar from A_k, the kth nonterminal.

Alternatively, element m_{ijk} can be viewed as the LD between an observed length-i substring of s,whose first symbol is b_j, and a prototype string y derivable from A_k. As an example, suppose A_k generates a string cd and that s = dcded. Two deletions are then required from the length-4 substring cded to change it to the string cd, which can be derived from A_k. In this case, $m_{4,2,k} = 2$.

A_r is arbitrarily chosen from the nonterminals to represent the start symbol, so that the element $m_{\ell 1 r}$ is zero if, and only if, the string s can be generated by the grammar.

Before proceeding to present a method for computing the M-matrix, various types of bielement rules are discussed.

Definition 4.5 The hierarchy level (HL) of a nonterminal A_k, denoted by H(k), is the number of symbols in a string derivable from that nonterminal.

In general, the HL of a nonterminal in a CFG is not unique. However, because of the nature of the construction procedure described later, it is guaranteed that every nonterminal in the CFG's to be considered will have a unique-value hierarchy level, except for the start symbol A_r, whose HL may be multi-valued. The HL of A_r is, of course, equal to ℓ, the length of the string under consideration.

By definition, the HL of any nonterminal in a terminating rule is unity for the CFG's inferred in this chapter.

In order for the elements of the M-matrix to have the meaning given above, it is necessary for nonterminals in bielement rules of the form $A_k \longrightarrow A_p A_q$, where A_k, A_p and $A_q \in V_N$, to have hierarchy levels satisfying at least one of the following conditions :-

$$H(k) = H(p) + 1 \qquad (4.5)$$

$$H(k) = H(q) + 1 \qquad (4.6)$$

That is, the HL of A_k should differ from that of A_p or A_q by exactly one.

This condition excludes, from CFG's to be constructed, bielement rules of the form $A_k \longrightarrow A_p A_q$ where neither A_p nor A_q are in the terminating rules.

Permissible types of rules are where :-

(type 1) Both A_p and A_q are in terminating rules (4.7)

(type 2) A_p or A_q, but not both, are in the terminating rules (4.8)

The HL's of various nonterminals in a CFG can be computed as follows.

(a) HL's of nonterminals in terminating rules

$$H(k) = 1 \qquad \text{for all } A_k \longrightarrow a_k \text{ of the CFG.} \tag{4.9}$$

This follows from the definition of the HL.

(b) HL's of nonterminals in bielement rules

$$H(k) = \underset{p,q \in P_k}{\text{Min}} \left[H(p) + H(q) \right] \tag{4.10}$$

where P_k is the set of ordered pairs (p,q) such that $A_k \longrightarrow A_p A_q$ is a rule of the CFG.

The above follows because nonterminal A_k is replaced by nonterminals A_p and A_q via a bielement rule of the form $A_k \longrightarrow A_p A_q$, so that HL's corresponding to A_p and A_q need to be added. The final result is the smallest of this sum taken over all rules in the set P_k.

The following describes an iterative procedure for computing the M-matrix. The procedure is in two parts; the first part is for the terminating rules whilst the second is for the bielement rules.

Part 1 : Terminating rules

(i) $i = 1$

$$m_{1jk} = \begin{cases} 0, & \text{if and only if } A_k \longrightarrow b_j \text{ is a rule of the CFG} \\ 1, & \text{otherwise} \end{cases} \tag{4.11}$$

This follows from the definitions of M and of terminating rules.

(ii) $i = 2,3,\ldots,\ell$

$$m_{ijk} = \text{Max} \left[(i-1), \sum_{u=j}^{j+i-1} m_{1uk} \right] \tag{4.12}$$

The above follows because the number of alterations required for a substring of length i to be derivable from A_k is i-1, if A_k

generates any of the symbol in the substring, otherwise it is i.

Part 2 : Bielement rules

(iii) $i = 1$

$$m_{1jk} = \min_{p,q \in P_k} \left\{ \min \left[m_{1jp} + H(q),\ m_{1jq} + H(p) \right] \right\} \qquad (4.13)$$

where P_k is defined as before.

This follows because nonterminal A_k is replaced by nonterminals A_p and A_q via a bielement rule of the form $A_k \longrightarrow A_p A_q$, so that the element of the M-matrix and HL corresponding to A_p and A_q respectively, or vice versa, need to be added. The smallest of this sum is selected because the minimum number of alterations needed to convert a string, derivable from A_k, into a substring of length unity cannot be less than the HL of H(p) or H(q), whichever is smaller.

(iv) $i = 2,3,\ldots,\ell$

$$m_{ijk} = \min_{p,q \in P_k} \left\{ \min \left[m_{ijp} + H(q),\ m_{ijq} + H(p),\ \min_{1 \leq u < i} \left(m_{ujp} + m_{i-u,j+u,q} \right) \right] \right\} \qquad (4.14)$$

where P_k is defined as before.

The above follows in a way similar to the preceding case, with the extra consideration that the substring of length i is itself divided into two sub-substrings of lengths u and i-u, for $u = 1,\ldots,i-1$. The first of these sub-substrings starts at the jth symbol and the second at the (j+u)th symbol.

A unit length string generated by the grammar can be represented by a rule $A_r \longrightarrow A_p A_q$ where either A_p or A_q is arbitrarily chosen to act as a 'dummy' nonterminal. That is, the chosen dummy

nonterminal never appears in any other rules of the grammar, except in the specified start rule. The other nonterminal is, of course, in a terminating rule. This is done in order to preserve the format of the normal form of the CFG. The contribution of such a rule to the content of the element of the M-matrix corresponding to LS nonterminal, ie. A_r, is equal to the content of the element of M corresponding to non-dummy nonterminal (either A_p or A_q) .

It is possible to achieve some savings in the computation of the M-matrix. This is because not all elements of the M-matrix are required to be computed. Elements m_{ijk} for which $i+j > \ell+1$ need not be computed, as there are no substrings of s corresponding to such values of i and j. In addition, provided that the length of s exceeds unity, elements for which i = 1 and which correspond to the start symbol, ie. k = r, need not be computed.

The following example illustrates a completely filled M-matrix for a given string and a specified CFG.

Example 4.2 Consider a normal form CFG $G_{4.2} = (V_N, \Sigma, R, £)$

where $V_N = (A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_{31}, A_{32}, A_{33}, A_r = £)$

$\Sigma = (B, C, D, E, e, j, h)$

$R =$

$A_1 \rightarrow e$	$A_{31} \rightarrow A_1 A_2$	$A_r \rightarrow A_{32} A_4$
$A_2 \rightarrow E$	$A_{32} \rightarrow A_{31} A_3$	$A_r \rightarrow A_5 A_{33}$
$A_3 \rightarrow h$	$A_{33} \rightarrow A_6 A_7$	
$A_4 \rightarrow D$		
$A_5 \rightarrow B$		
$A_6 \rightarrow j$		
$A_7 \rightarrow C$		

The M-matrix (nonweighted) for the string s = BjC and the

grammar $G_{4.2}$ is given in table 4.1 below.

i	1			2		3
j	1	2	3	1	2	1
substring	B	j	C	Bj	jC	BjC
A_1	1	1	1	2	2	3
A_2	1	1	1	2	2	3
A_3	1	1	1	2	2	3
A_4	1	1	1	2	2	3
A_5	0	1	1	1	2	2
A_6	1	0	1	1	1	2
A_7	1	1	0	2	1	2
A_{31}	2	2	2	2	2	3
A_{32}	3	3	3	3	3	3
A_{33}	2	1	1	2	0	1
A_r	-	-	-	1	1	0

Table 4.1 M-matrix for string s = BjC and grammar $G_{4.2}$

4.3.2 Weighted M-matrix

This section describes an iterative procedure for computing the weighted version of the M-matrix. This type of the M-matrix is usually employed if it is decided to attach some sort of 'significance' to each and every symbol in the alphabet. Not all parts of the iterative procedure for computing the nonweighted M-matrix need to be modified in order to obtain the weighted one. Thus, unless stated otherwise, all definitions and derivations required for the computation of the weighted M-matrix are assumed to be the same as those given in the previous section.

First, some definitions necessary for the discussion of the iterative procedure are given below.

Definition 4.6 For a given CFG in Chomsky normal form and for a string s as described in definition 4.4, the weighted M-matrix (hereafter referred to as wM-matrix), wM, is a three dimensional, $\ell * \ell * r$ matrix ; where ℓ and r have the same meanings as in definition 4.4 . Element m_{ijk} of wM denotes the WLD between an observed length-i substring of s, whose first symbol is b_j, and a prototype string y generated from A_k, the kth nonterminal.

Using the same example that exemplifies definition 4.4, where A_k derives cd and that s = dcded, the content of element $m_{4,2,k}$ of the wM-matrix associated with substring cded becomes |significance of 'e'| + |significance of 'd'| . Substituting for the significance values of symbols 'e' and 'd' as given by the table in appendix B yields $m_{4,2,k} = 5 + 4 = 9$.

Definition 4.7 The weighted hierarchy level (WHL) of a nonterminal A_k, denoted by wH(k), is the sum of the absolute values of the significance

of all symbols in a string derivable from A_k.

As before, every nonterminal in the CFG's to be inferred will have a unique-valued WHL, except for the start symbol A_r.

The following is a procedure for computing the WHL's of nonterminals in a CFG.

(a) WHL's of nonterminals in terminating rules

$$wH(k) = \left|\text{significance of } a_k\right| \text{ for a rule } A_k \longrightarrow a_k \text{ in the CFG} \tag{4.15}$$

This, again, follows from the definition of the WHL.

(b) WHL's of nonterminals in bielement rules

$$wH(k) = \min_{p,q \in P_k} \left[wH(p) + wH(q)\right] \tag{4.16}$$

where P_k is defined as before.

The above follows from the same reasons given for equation 4.10 .

An iterative procedure for the computation of the wM-matrix to be presented below is also divided into two parts, one each for the terminating and bielement rules. The part for bielement rules is the same as part 2 of the procedure for the computation of the M-matrix, except that all the HL's are replaced by their corresponding WHL's.

Part 1 : Terminating rules

(i) $i = 1$

$$m_{1jk} = \left|\text{significance of } b_j - \text{significance of } a_k\right| \quad \text{for a rule } A_k \longrightarrow a_k \text{ in the CFG.} \tag{4.17}$$

This follows from the definitions of wM-matrix and of terminating rules.

(ii) $i = 2,3,\ldots,\ell$

$$m_{ijk} = \min_{u=j}^{j+i-1} \left\{ m_{luk} + \sum_{v=1}^{\ell} \left|\text{significance of } b_v\right| - \left|\text{significance of } b_u\right| \right\} \qquad (4.18)$$

The above follows because a substring of length-i, whose first symbol starts at position j can be considered as to consist of i sub-substrings, each of unit length. Only one of these sub-substrings can be matched against a_k, the terminal in a rule $A_k \longrightarrow a_k$ of the CFG.

Part 2 : Bielement rules

(iii) $i = 1$

$$m_{1jk} = \min_{p,q \in P_k} \left\{ \text{Min}\left[m_{1jp} + wH(q),\ m_{1jq} + wH(p) \right] \right\} \qquad (4.19)$$

where P_k is defined as before.

This follows from the same reasons as those given for equation 4.13 with the replacement of H(p) and H(q) by wH(p) and wH(q) respectively.

(iv) $i = 2,3,\ldots,\ell$

$$m_{ijk} = \min_{p,q \in P_k} \left\{ \text{Min}\left[m_{ijp} + wH(q),\ m_{ijq} + wH(p),\ \min_{1 \le u < i} \left(m_{ujp} + m_{i-u,j+u,q} \right) \right] \right\} \qquad (4.20)$$

where P_k is defined as before.

This also follows from the same reasons that elucidate equation 4.14, with H(p) and H(q) being replaced by wH(p) and wH(q) respectively.

An example of a wM-matrix is illustrated by table 4.2. The string is s = BjC and the grammar is the CFG $G_{4.2}$ taken from example 4.2. Values of significance of various symbols can be found in appendix B.

i	1			2		3
j	1	2	3	1	2	1
substring	B	j	C	Bj	jC	BjC
A_1	7	5	8	7	8	10
A_2	3	15	2	13	12	14
A_3	10	2	11	4	5	7
A_4	2	14	1	12	11	13
A_5	0	12	1	10	11	13
A_6	12	0	13	2	3	5
A_7	1	13	0	11	10	12
A_{31}	8	10	7	12	7	9
A_{32}	16	12	15	10	15	13
A_{33}	11	3	10	5	0	2
A_r	-	-	-	3	2	0

Table 4.2 wM-matrix for a string s = BjC and the grammar $G_{4.2}$

4.4 Inference of a CFG

The inference method presented in this section is based on a search for incompatibility between each string in the sample set and the current grammar. For each occurrence of such incompatibility, the grammar is augmented such that a new grammar is produced which can generate the present string. Fig. 4.2 illustrates the overall structure of the inference method which can be explained as follows.

The symbol strings in the sample set S_+ are assumed to be arbitrarily labelled $s_1, s_2, \ldots, s_{M_S}$. The first step is to select the required type of the M-matrix (weighted or otherwise). An initial CFG is constructed from the first string s_1 such that G_1 generates exactly that string ie. G_1 is a SG. If there exists only one string in S_+, the required grammar is G_1. Otherwise, the inference method is applied recursively as follows. The nth string, s_n, is matched against the (n-1)th inferred CFG G_{n-1}, for $n = 2, \ldots, M_S$. The matching process involves the computation of the appropriate M-matrix whose elements reveal the shortcomings of the CFG in relation to its ability to generate the string. If the element $m_{\ell 1 r} = 0$, s_n is derivable from G_{n-1} and no change is required, ie. $G_n = G_{n-1}$. Otherwise, information from the particular M-matrix is used to augment G_{n-1} by appending additional terminals, nonterminals and rules, as appropriate, so that the new grammar G_n can generate the string. The above method is repeated until all strings in S_+ have been processed.

The inference method just described will be presented in two stages. The first stage is concerned with the creation of the initial set of productions from the first string in the sample set. In the second stage of presentation, the selected type of the M-matrix is computed using the procedures given in the previous section. The current

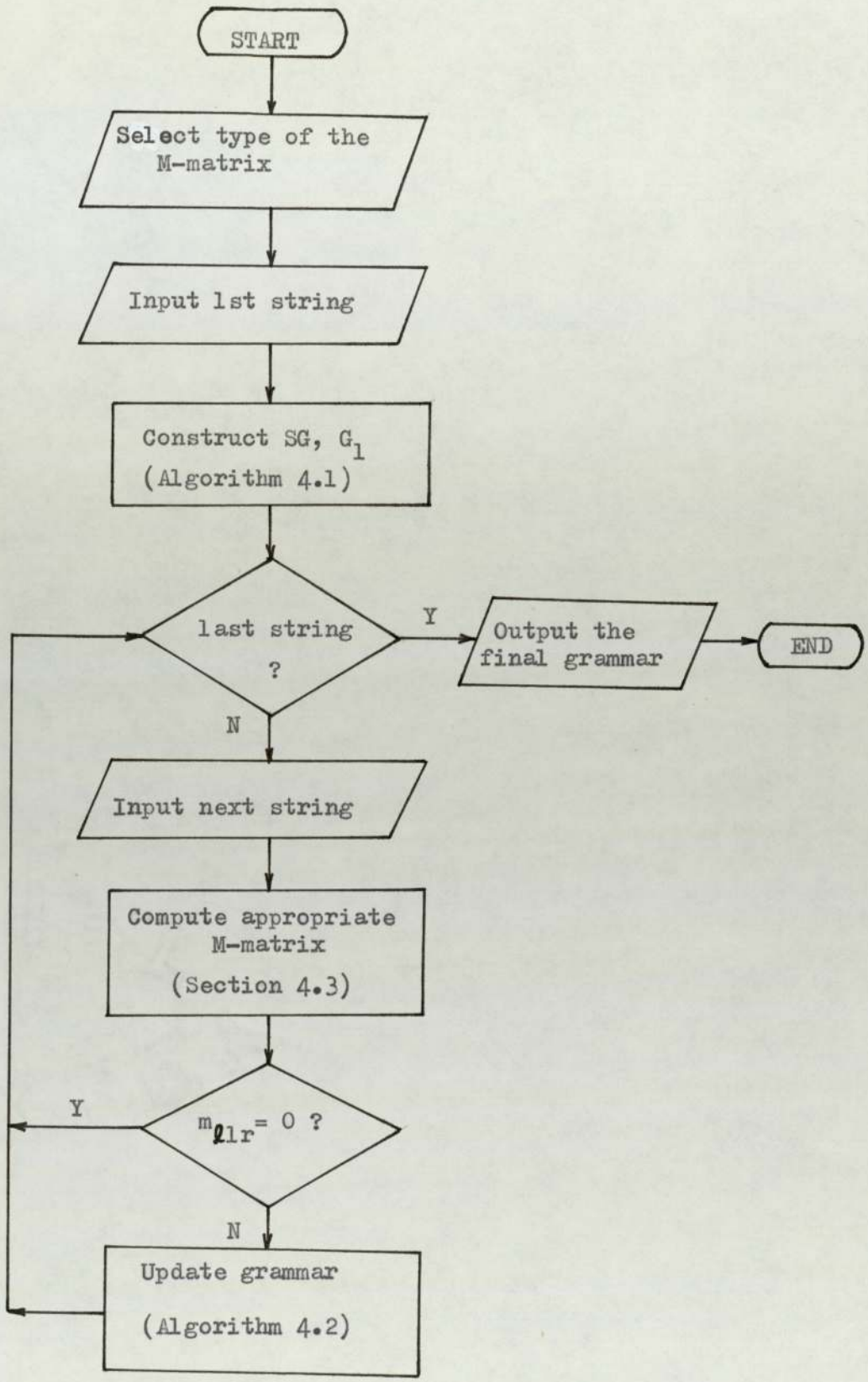


Fig. 4.2 A schematic diagram of an inference method of a CFG

grammar is then updated where necessary to form an augmented grammar as required.

4.4.1 Formulation of the initial set of rewriting rules

In the first stage of the inference method, it is arbitrarily chosen to process the first string from left to right starting from the left most symbol of the string. A set of terminating rules is formed first. Then a bielement rule of type 1 is constructed from two non-terminals corresponding to the first two symbols of the string. This is followed by the formulation of successive bielement rules of type 2, where only A_q is in a terminating rule, until the entire string is dealt with.

An algorithm for forming the initial CFG is now given.

Algorithm 4.1

Step 1 Read the first string $s_1 = b_1 b_2 \ldots b_\ell$ where s_1 is arbitrarily drawn from the sample set.

Step 2 Form a set of terminating rules as follows.

For i = 1 to ℓ:

Unless a rule has already been formed with b_i on its RS, create a new nonterminal A_{b_i} and a new terminating rule

$$A_{b_i} \longrightarrow b_i \ .$$

(The notation introduced here indicates that A_{b_i} is the nonterminal corresponding to b_i) .

The nonterminals derived above form the set of nonterminals having unity hierarchy level.

Step 3 Create a new nonterminal of hierarchy level 2 and a bielement rule of type 1 from the nonterminals corresponding to the first two symbols of s_1: $A_{H_2} \longrightarrow A_{b_1} A_{b_2}$

(A_{H_i} denotes a nonterminal with hierarchy level i) .

If $\ell = 2$, A_{H_2} ($= A_r$) is the start symbol, and the formation of G_1 is complete.

Otherwise, further bielement rules of type 2 are formed as follows.

For i = 3 to ℓ:

Create a new nonterminal A_{H_i} and a new bielement rule:

$A_{H_i} \longrightarrow A_{H_{i-1}} A_{b_i}$.

A_{H_ℓ} ($=A_r$) is the start symbol. This completes the construction of G_1. The corresponding schematic diagram is depicted in Fig. 4.3 .

4.4.2 Updating the existing grammar

The following algorithm is employed in the formulation of G_n from G_{n-1}, for $n = 2,\ldots,M_S$.

Algorithm 4.2

Step 1 Read a string $s_n = b_1 b_2 \ldots b_\ell$.

Step 2 Find iteratively, using the procedures in section 4.3, all necessary entries of the chosen type of M-matrix for s_n.

Step 3 If $m_{\ell 1 r}$ is not zero, go to step 4.

Otherwise, $G_n = G_{n-1}$.

If s_n is the last string, END.

Otherwise, increase n by one and go to step 1.

Step 4 Formation of terminating rules

For i = 1 to ℓ :

Create a new nonterminal A_{b_i} and a new terminating rule

$A_{b_i} \longrightarrow b_i$ if, and only if, there does not exist a rule with b_i at its RS.

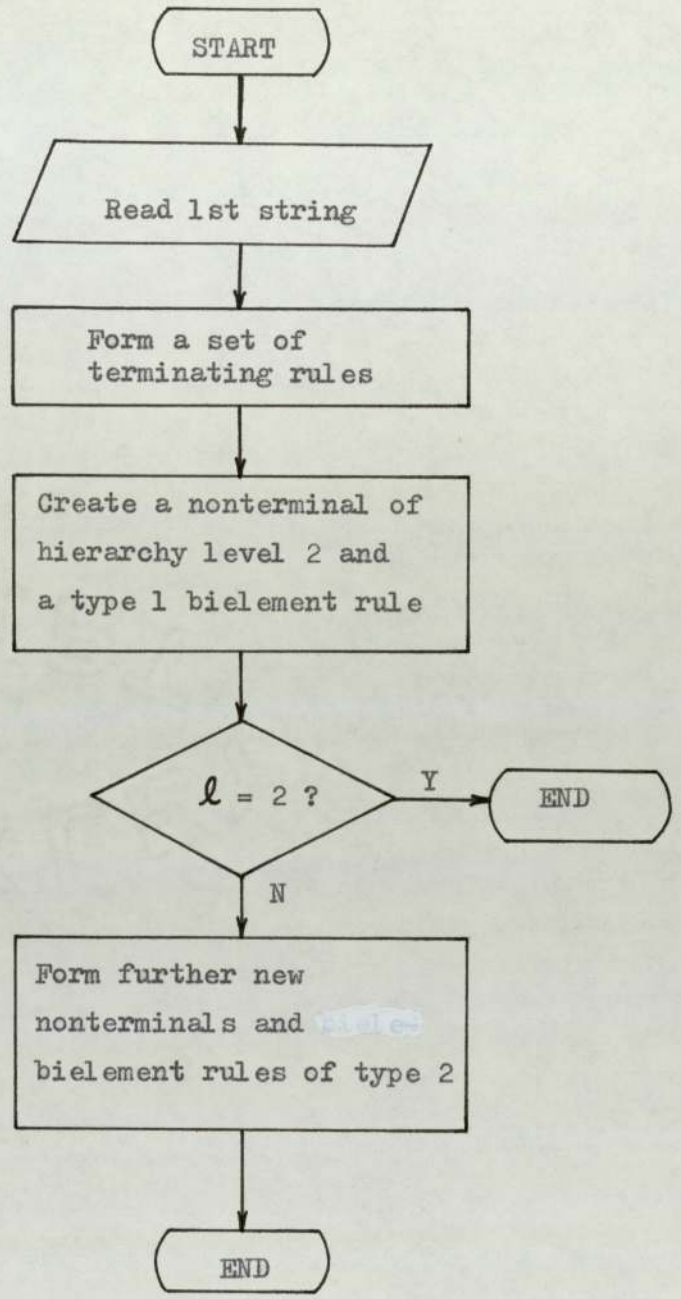


Fig. 4.3 A schematic diagram of algorithm 4.1

Formation of bielement rules

Step 5 Select a set of indices j_i :

For $i = \ell - 1$:

Select $j_{\ell-1}$ as the least j for which $m_{\ell-1,j,r}$ is minimum.

For $i = (\ell-2), (\ell-3), \ldots 2$:

Select j_i as the least j for which m_{ijr} is minimum and for which $j_i \geqslant j_{i+1}$

Each j_i is the j index of m_{ijk} corresponding to a substring of length i.

Each will lie in the range $1 \leqslant j_i \leqslant \ell - i + 1$ because, as explained previously in section 4.3.1, there are no substrings for values outside this range.

Step 6 Form the new bielement rules.

For $i = 2$:

Create a new nonterminal A_{h_2} and a new rule $A_{h_2} \longrightarrow A_{j_2} A_{j_2+1}$ unless these nonterminals and the rule have already been created. A_{j_2} and A_{j_2+1} are the nonterminals in the terminating rules having b_{j_2} and b_{j_2+1}, respectively, on the RS.

For $i = 3$ to ℓ :

If $j_{i-1} > j_i$, form a new bielement rule $A_{h_i} \longrightarrow A_{j_i} A_{h_i-1}$, where A_{h_i} is a newly created nonterminal, unless there is already a rule of the form $A_q \longrightarrow BA_{h_i-1}$, in which case A_q is used as A_{h_i}. (B represents an arbitrary nonterminal).

If $j_{i-1} = j_i$, form a new bielement rule $A_{h_i} \longrightarrow A_{h_i-1} A_{j_i+H(h_i-1)}$, where A_{h_i} is a newly created nonterminal, unless there is already a rule of the form

$A_q \longrightarrow A_{h_i-1}B$, in which case A_q is used as A_{h_i}.

Represent A_{h_ℓ} as A_r, the start symbol.

If s_n is the last string, END.

Otherwise, increase n by one and go to step 1.

A schematic diagram of the above algorithm is depicted in Fig. 4.4 .

4.5 Illustrative example of a CFG inference

The following set of sample strings, taken from Bezdel and Bridle[20], represents the output from the FE in a speech recognition system when the word 'SEVEN' was spoken by different speakers.

The strings are :

s_1 = sauau

s_2 = fsau

s_3 = fsu

s_4 = saiaua

s_5 = sau

s_6 = fpsau

s_7 = saiau

The following illustrates the step-by-step operation of the inference method using the nonweighted M-matrix. Only the elements of the M-matrix relevant to the augmentation of the CFG's are shown, ie., the entries m_{ijr} for i =2,...,ℓ and j = 1,..., ℓ+1-i. The indices j_i selected are indicated by a prime on the associated element of M. The rules added at a given stage are indicated by the use of $\sim$ lines underneath the rules.

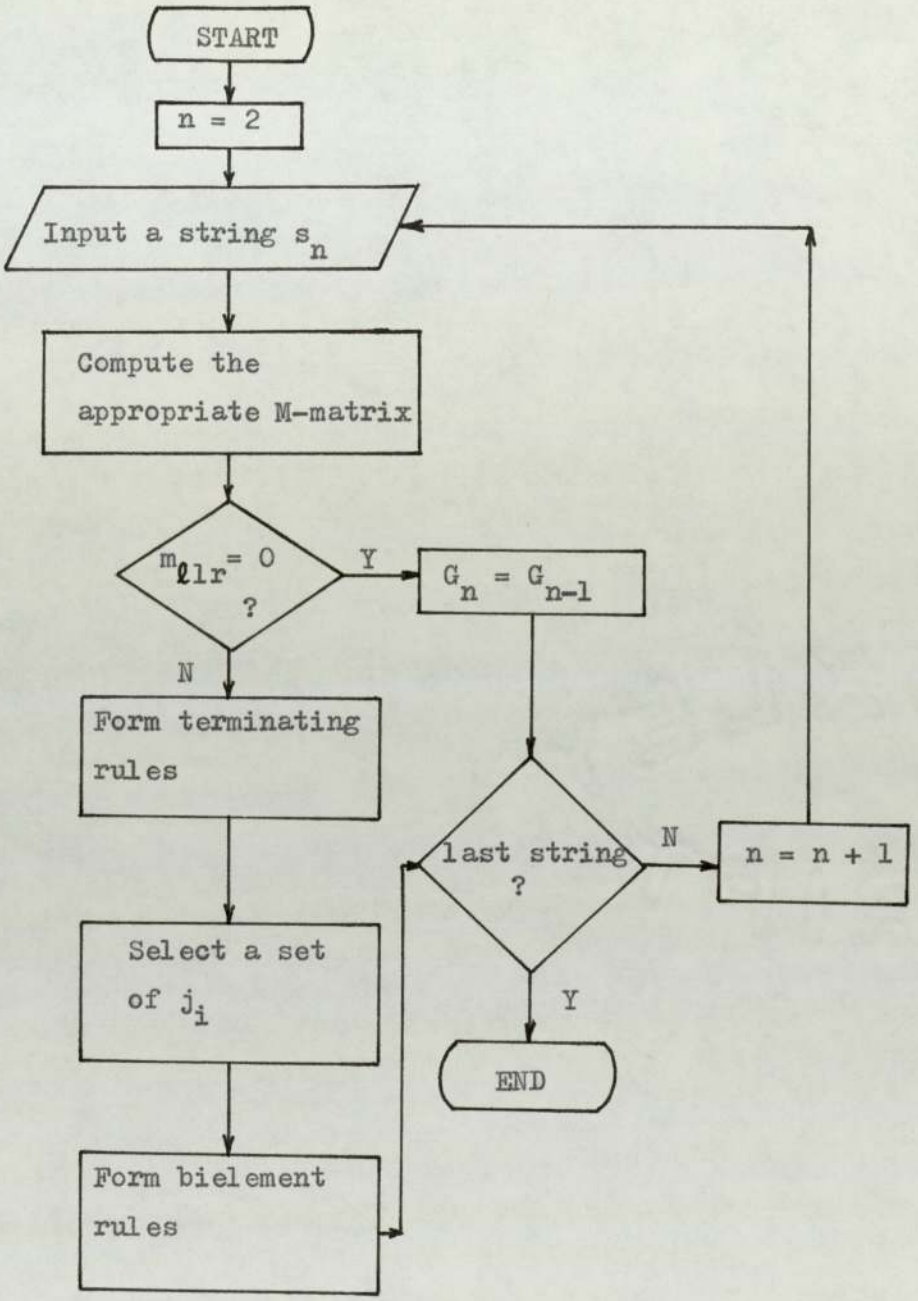


Fig. 4.4 A schematic diagram illustrating an algorithm for updating a CFG

Stage 1

Formation of G_1 directly from s_1 .

s_1 = sauau

G_1: $A_1 \rightarrow s$ $A_2 \rightarrow a$ $A_3 \rightarrow u$ $A_{12} \rightarrow A_1A_2$ $A_{13} \rightarrow A_{12}A_3$ $A_{14} \rightarrow A_{13}A_2$ $A_r \rightarrow A_{14}A_3$

Stage 2

s_2 = fsau

G_2: $A_1 \rightarrow s$ $A_2 \rightarrow a$ $A_3 \rightarrow u$ $A_4 \rightarrow f$ $A_{12} \rightarrow A_1A_2$ $A_{13} \rightarrow A_{12}A_3$ $A_{14} \rightarrow A_{13}A_2$ $A_r \rightarrow A_{14}A_3$ $A_r \rightarrow A_4A_{13}$

M=

i \ j	1	2	3
2	5	3'	3
3	4	2'	
4	3		

Stage 3

s_3 = fsu

G_3: $A_1 \rightarrow s$ $A_2 \rightarrow a$ $A_3 \rightarrow u$ $A_4 \rightarrow f$ $A_{12} \rightarrow A_1A_2$ $A_{13} \rightarrow A_{12}A_3$ $A_{14} \rightarrow A_{13}A_2$ $A_{15} \rightarrow A_4A_1$ $A_r \rightarrow A_{14}A_3$ $A_r \rightarrow A_4A_{13}$ $A_r \rightarrow A_{15}A_3$

M=

i \ j	1	2
2	2'	2
3	1	

Stage 4

s_4 = saiaua

G_4: $A_1 \rightarrow s$ $A_2 \rightarrow a$ $A_3 \rightarrow u$ $A_4 \rightarrow f$ $A_5 \rightarrow i$ $A_{12} \rightarrow A_1A_2$ $A_{13} \rightarrow A_{12}A_3$ $A_{13} \rightarrow A_{12}A_5$ $A_{14} \rightarrow A_{13}A_2$ $A_{15} \rightarrow A_4A_1$ $A_{16} \rightarrow A_{14}A_3$ $A_r \rightarrow A_{14}A_3$ $A_r \rightarrow A_4A_{13}$ $A_r \rightarrow A_{15}A_3$ $A_r \rightarrow A_{16}A_2$

M=

i \ j	1	2	3	4	5
2	2'	3	3	2	3
3	2'	3	2	2	
4	2'	2	2		
5	1'	3			
6	2				

At the end of this stage, G_4 predicts three additional strings, namely, sauaua, saiau, and fsai.

Note that the predicted string saiau subsequently appears in the set as s_7.

Stage 5

s_5 = sau

M=

i \ j	1	2
2	2'	2
3	1	

G_5: $A_1 \rightarrow s$, $A_2 \rightarrow a$, $A_3 \rightarrow u$, $A_4 \rightarrow f$, $A_5 \rightarrow i$, $A_{12} \rightarrow A_1A_2$, $A_{13} \rightarrow A_{12}A_3$, $A_{13} \rightarrow A_{12}A_5$, $A_{14} \rightarrow A_{13}A_2$, $A_{15} \rightarrow A_4A_1$, $A_{16} \rightarrow A_{14}A_3$, $A_r \rightarrow A_{14}A_3$, $A_r \rightarrow A_4A_{13}$, $A_r \rightarrow A_{15}A_3$, $A_r \rightarrow A_{16}A_2$, $A_r \rightarrow A_{12}A_3$

Stage 6

s_6 = fpsau

M=

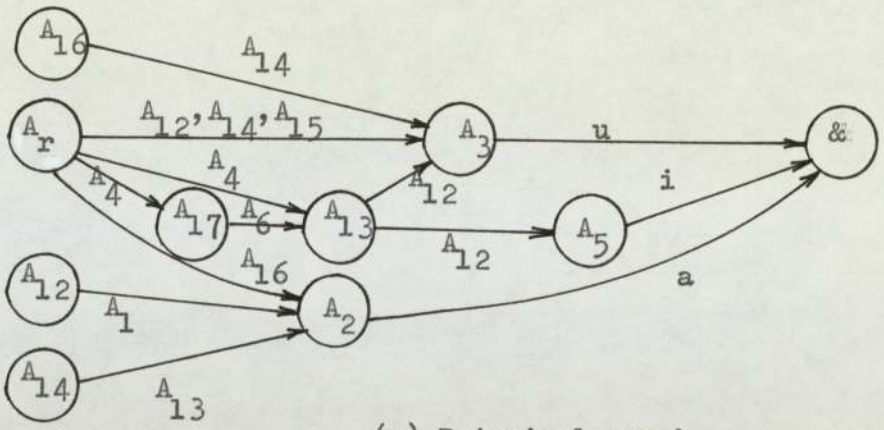
i \ j	1	2	3	4
2	2	2	1'	1
3	2	2	0'	
4	2	1'		
5	1			

G_6: $A_1 \rightarrow s$, $A_2 \rightarrow a$, $A_3 \rightarrow u$, $A_4 \rightarrow f$, $A_5 \rightarrow i$, $A_6 \rightarrow p$, $A_{12} \rightarrow A_1A_2$, $A_{13} \rightarrow A_{12}A_3$, $A_{13} \rightarrow A_{12}A_5$, $A_{14} \rightarrow A_{13}A_2$, $A_{15} \rightarrow A_4A_1$, $A_{16} \rightarrow A_{14}A_3$, $A_{17} \rightarrow A_6A_{13}$, $A_r \rightarrow A_{14}A_3$, $A_r \rightarrow A_4A_{13}$, $A_r \rightarrow A_{15}A_3$, $A_r \rightarrow A_{16}A_2$, $A_r \rightarrow A_{12}A_3$, $A_r \rightarrow A_4A_{17}$

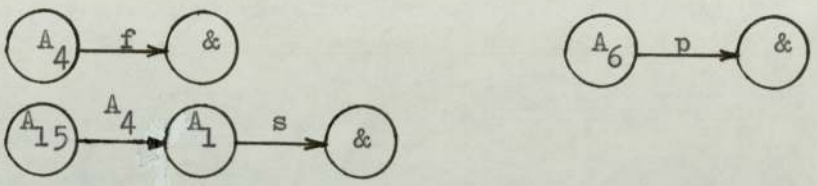
The string fpsai is added to the set by G_6.

Since s_7 = saiau is already in the set, the final grammar is G_6.

A PTN of G_6 is depicted in Fig. 4.5 .



(a) Principal graph



(b) auxiliary graphs

Fig. 4.5 A PTN of the inferred grammar G_6

4.6 A recognition scheme for CFG models

Fig. 4.6 displays the general features of the recognition scheme B which is another method of the recognition of isolated words. Two major differences distinguish scheme B from scheme A described in section 3.3 and Fig. 3.13. The first difference is that, in scheme B, production probabilities are employed to select the most likely word in both of the following cases : (i) an exact match - where two or more grammars can generate the observed string, and (ii) a closest match - where two or more grammars could nearly have generated the string with the same penalty incurred. The second difference is concerned with the use of the AWSL criterion (to be explained later) in place of a stochastic method in scheme A.

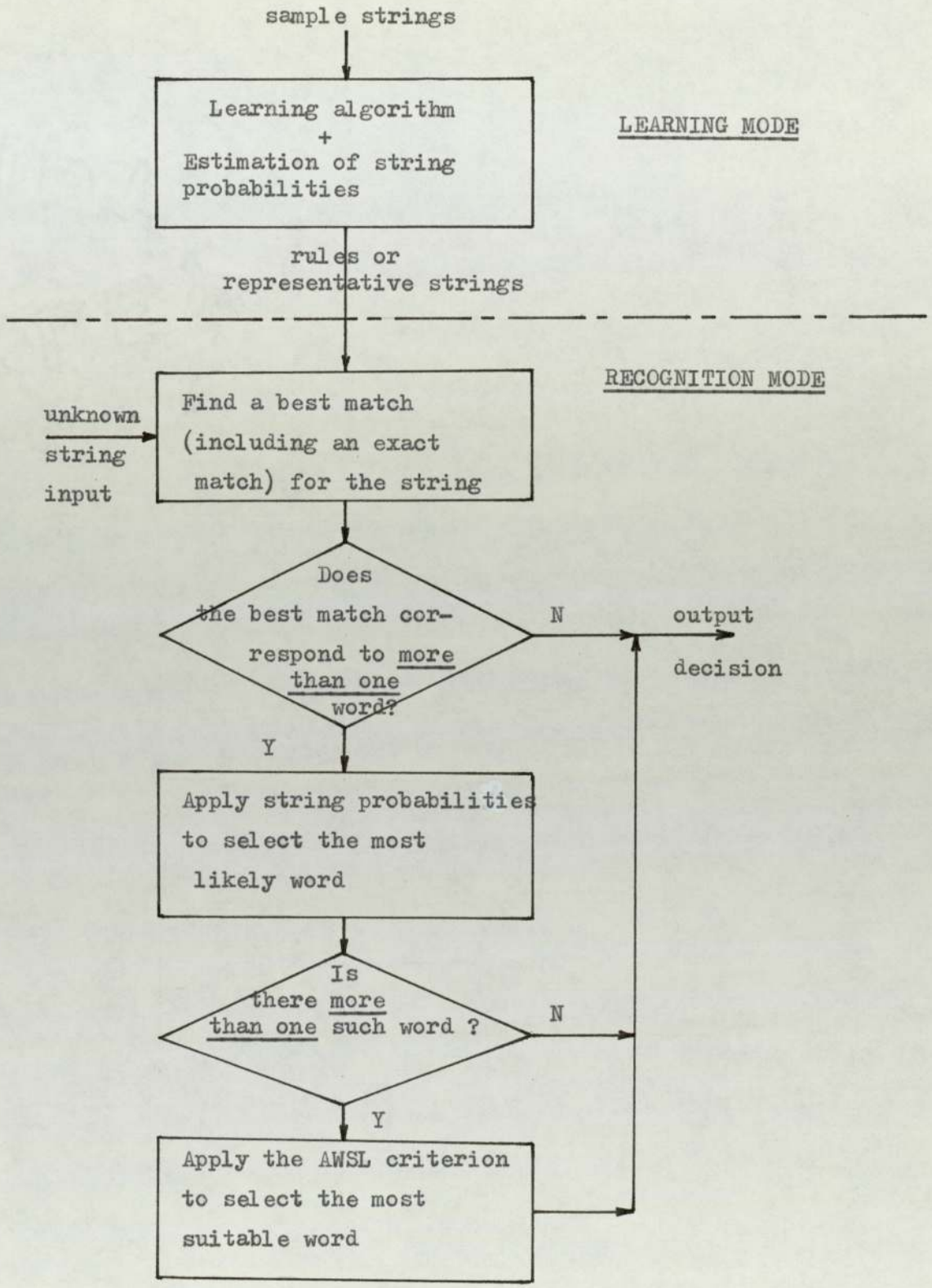


Fig. 4.6 Flow diagram of the recognition scheme B

The following outlines a recognition system whose flow diagram is shown in Fig. 4.7 based on the above mentioned recognition scheme B and pre-inferred CFG's.

As in scheme A (and in many IWR systems), the recognition system consists of two phases of operation - the learning phase followed by the recognition process. In the learning mode, normal form CFG's, one for each word in the vocabulary, are directly constructed from a set of sample strings using the method of section 4.4 . The inference process involves the computation of either a wM or an M-matrix whose elements can be iteratively computed by the procedures given in section 4.3 . The prior knowledge of the significance of symbols involved (or the lack of it) influences the selection of the type of the M-matrix. Estimation of production probabilities of the inferred CFG's is also carried out during the learning operation.

In the recognition mode, an incoming string is analysed to determine which grammar, if any, could have generated it. The determination of a best match for the string, which can be either an exact match or a closest match, is accomplished by using the wM-matrix as a recognition matrix. The foregoing statement assumes, of course, that the significance of various symbols in the alphabet is known or can be determined beforehand. The wM rather than the M-matrix is chosen because, from experimental observations, the recognition performance when employing the former improves significantly over that when the latter is used.

Appropriate decision is given at the output of the system if the best match found above corresponds to only one word in the vocabulary. Otherwise, production probabilities are employed to select the word that is the most likely (probabilistically) to have correspond

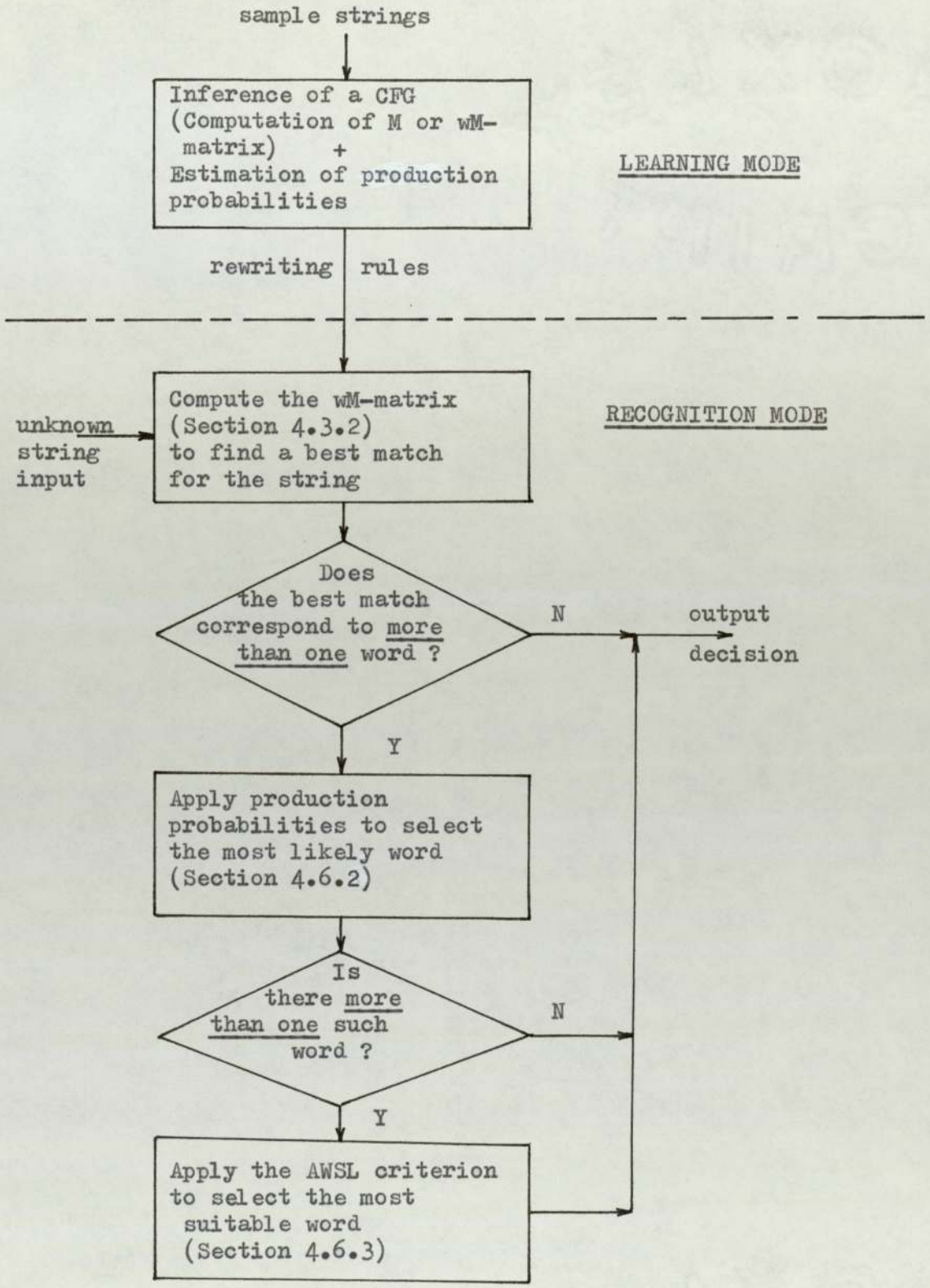


Fig. 4.7 A CFG-based recognition system using scheme B of the recognition method

to the best match. For the case where there occur two or more such equally likely words, a selection is made of the most suitable word according to the AWSL criterion.

In the following sections, descriptions are given of various recognition operations mentioned above. These will be followed by the formal presentation of a recognition algorithm comprising the fore-mentioned methods for the representation of the overall recognition process.

4.6.1 The wM-matrix as a recognition matrix

The determination of structures or syntactic analysis of strings generated by CFG's have been studied and investigated by many researchers in the computing field. Numerous algorithms have been proposed for the recognition of CFL's, for example, those in references 89-94. Among the algorithms mentioned above, that of Younger[94] is similar to the one presented here in the format of presentation. Major features of each method can be described as follows. In the recognition matrix of Younger, an incoming string is accepted as belonging to the language of a given CFG provided a certain element of this matrix is 1. If the element of the matrix is zero, the string is rejected. In the method to be given below, a certain element of the wM-matrix represents the smallest distance (WLD) between an input string x and some string y generated by the given CFG. In other words, the string x is parsed to completion on the basis of minimizing the number of syntax errors or symbol alterations. If the content of this element of the wM-matrix is zero, x becomes an exact match of y.

Descriptions are now given of how to apply the wM-matrix in the recognition of isolated words. First, assume that all CFG's

associated with each word in the vocabulary are in, or have been reduced to Chomsky normal form. A recognition can then be performed on an input string s of length ℓ as follows. Form a wM-matrix for the string s and each of the CFG using the procedure described in section 4.3.2 . Decide that the string s corresponds to word w_k if, and only if, the element $m(\ell,1,r)$ of the wM-matrix associated with word w_k is the largest for all such elements corresponding to all words in the vocabulary, where A_r is the start symbol.

The above follows immediately from the definition of the wM-matrix. In essence, the method is concerned with the determination of the smallest WLD between the string s and some strings derivable from each of the CFG's under consideration. For the case where there occur two or more words associated with the same value of WLD, the technique of the next section is applied to select the most probable word.

The foregoing method also, of course, works with the M-matrix (ie. nonweighted version). Computation of the required M-matrix is accomplished via the appropriate application of the procedure of section 4.3.1 . In this case, the element $m(\ell,1,r)$ of the M-matrix denotes the LD between the string s and some string y derivable from the given CFG.

Although, both types of the M-matrix can be employed in the recognition of isolated words, the weighted version is preferable to the nonweighted one. This is because the use of the former as a recognition matrix considerably reduces the occurrence of the situations where two or more grammars can generate the string s with the same minimum number of alterations . The above is hardly surprising since more information about the strings is available to the weighted type

of the M-matrix than that available to its counterpart.

A parse (or parses) for the string s can be readily determined from either the wM-matrix or the M-matrix in the following manners. For the string s to be accepted by a given CFG, all entries of the M-matrix (or the wM-matrix) associating with a parse of s must be zero. This follows because in order for the entry $m(\ell,1,r)$ to become zero, each nonterminal in the rules that are employed in the derivation of s must contribute exactly zero alteration to its associated element of the M-matrix. These zero entries can then be used to construct a parse for that string as illustrated by the following example.

Example 4.3 Consider a string s = uau and a given normal form CFG

$G_{4.3} = (V_N, \Sigma, R, \pounds)$ where :-

$V_N = (A_1, A_2, A_3, A_{12}, A_{13}, A_{14}, A_{15}, A_r = \pounds)$

$\Sigma = (a, i, u)$

$R = A_1 \rightarrow u \quad A_{12} \rightarrow A_1A_2 \quad A_r \rightarrow A_{12}A_1$

$A_2 \rightarrow a \quad A_{13} \rightarrow A_2A_1 \quad A_r \rightarrow A_2A_1$

$A_3 \rightarrow i \quad A_{14} \rightarrow A_1A_{13} \quad A_r \rightarrow A_{15}A_1$

$A_{15} \rightarrow A_{14}A_3 \quad A_r \rightarrow A_{14}A_2$

$A_r \rightarrow A_{14}A_3$

The M-matrix of s with respect to $G_{4.3}$ is shown in table 4.3 .

i	1			2		3
j	1	2	3	1	2	1
substring	u	a	u	ua	au	uau
A_1	$\underline{0}$	1	$\underline{0}$	1	1	2
A_2	1	$\underline{0}$	1	1	1	2
A_3	1	1	1	2	2	3
A_{12}	1	1	1	$\underline{0}$	2	1
A_{13}	1	1	1	2	0	1
A_{14}	2	2	2	1	1	0
A_{15}	3	3	3	2	2	1
A_r	-	-	-	1	0	$\underline{0}$

Table 4.3 The M-matrix for string s and grammar $G_{4.3}$

Since the nonterminal A_{13} does not appear in any of the start rewriting rules, the entry m(2,2,13), even though its value is zero, cannot be considered in the construction of the parse of s. Although the nonterminal A_{14} is in a start rewriting rule, that rule does not contribute zero penalty to the element m(3,1,r), and therefore A_{14} is not valid for the determination of the parse of s. A similar argument applies to element m(2,2,r). All valid zero entries in table 4.3 are shown underlined. Fig. 4.8 depicts the one and only one parse of s.

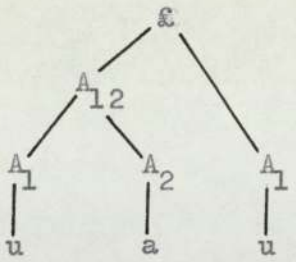


Fig. 4.8 A parse of string s = uau w.r.t. grammar $G_{4.3}$

4.6.2 Selection of the most likely word

This section describes a method for dealing with the situation mentioned in the previous section where the best match between an incoming string s and a given set of CFG's corresponds to two or more words in the vocabulary. For the case of an exact match, the above situation becomes, of course, the NE. The method presented here is most closely related to that of section 3.4.2 in the following way. Both methods are based on the approach of using stochastic grammars to determine the importance of various strings in probabilistic terms. The approach involves the counting of the frequency of usage of rules of the CFG's. However, the method given below is applied not only to cases of exact match, as is the method of section 3.4.2, but also to those of closest match.

The following definition follows closely that of a SFSG stated in definition 3.7 .

Definition 4.8 A stochastic normal form context-free grammar (SNCFG), G_{ns} is defined as :-

$$G_{ns} = (V_N, \Sigma, R_{ns}, £) \qquad (4.21)$$

where V_N, Σ, and £ are as defined earlier.

R_{ns} is a finite set of normal form stochastic productions, each of the form

$$A_i \xrightarrow{p_{ij}} B_j C_j \qquad A_i, B_j, C_j \in V_N$$

or $$A_k \xrightarrow{p_{kj}} a_j \qquad A_k \in \{V_N - \pounds\} \ , \ a_j \in \Sigma$$

where p_{ij} and p_{kj}, the production probabilities, are as defined in equations 3.2 and 3.3 , respectively.

Estimation of the above production probabilities follows the method given in section 3.4.2 .

Discussion is now presented concerning the forementioned method of selecting the most likely word. The basis of the approach which is based on the framework of Khert[95] on the investigation of the entropy of CFL's can be explained as follows.

It is of a normal practice to assume that the estimated probabilities associated with the productions of the SNCFG are independent. Given also that an input string s can be matched nearest to some string $y \in L(G_{ns})$ which can be generated from J distinctively different derivations in G_{ns}. It follows from the independence of the productions that the probability of generating y by one of the J derivations is equal to the product of the probabilities of sequence of productions employed in that derivation. The sum of the probability of each of these J derivations gives the overall probability associated with y. The method is then to apply the above procedure to each grammar that generates some string y having the same smallest value of WLD from the string s. Word w_k is selected as the most likely word whose grammar could most nearly have generated s if the above probability of string y corresponding to word w_k is the largest for all the words associated

with the same WLD.

4.6.3 The AWSL criterion

As mentioned earlier, the application of the method given in the last section to the recognition of a string s can result in the occurrence of two or more equally likely words. In such a situation, it is necessary to employ the technique given below to select only one word that is the most suitable according to a given criterion. Before proceeding with the presentation of the method, some definitions are first introduced as follows.

Definition 4.9 The average weighted string length (AWSL) for a given CFG is the sum of the absolute value of the significance of each and every symbol appearing in all strings in the sample set that has been used to infer that grammar, divided by the number of total sample strings.

Mathematically expressed, the AWSL can be calculated in the following manner.

Let the sample set be

$$S_+ = (s_j \mid j = 1,2,\ldots,M_S) \tag{4.22}$$

where $s_j = b_{j_1} b_{j_2} \ldots b_{j_\ell}$ is the jth string in S_+

M_S is the number of total strings in S_+

ℓ is the length of string s_j .

From the above definition,

$$\text{AWSL} = \frac{1}{M_S} \sum_{j=1}^{M_S} \sum_{i=1}^{\ell} \left| \text{significance of } b_{j_i} \right| \tag{4.23}$$

Definition 4.10 The weighted string length (WSL) of a string s is the sum of the absolute value of the significance of all symbols in s.

ie. $$\text{WSL of } s = \sum_{i=1}^{\ell} \left| \text{significance of } b_i \right| \qquad (4.24)$$

where $s = b_1 b_2 \ldots b_\ell$ for a string s of length ℓ.

The method is to compute the AWSL for CFG's associated with every word in the vocabulary. For an input string s, select word w_k as the most suitable word, provided the WSL of s is closest to the AWSL for the CFG corresponding to word w_k in comparison with all words in the vocabulary.

The foregoing procedure is based on experimental observations of various sets of strings corresponding to different words in the vocabulary. It is found that values of AWSL's for different sets of strings are reasonably placed from one another provided the number of strings in each set is not too small. The above technique based on the AWSL criterion thus provides a quick, simple and reasonably reliable method for solving the uncertainty situations such as those where there occur two or more equally likely words. Since the procedure is applied as the final stage, rather than as any of the earlier stages of the recognition process, the method can only improve the overall recognition performance and not impair it. It can be seen that the above is so if it is realized that the method is only applied when the output decision has to be made on two or more equally likely words. In this situation and without applying the AWSL criterion, it would not be possible to select one word from many equally probable alternatives, except for the arbitrary selection or guessing of the output. The result, in the worst case, with the inclusion of the criterion in the recognition process would be the same as above, when it is not included.

4.6.4 A recognition algorithm

The following is a formal presentation of an algorithm that is employed in the recognition system of Fig. 4.7 for the recognition of isolated words. All three methods previously described are incorporated in the algorithm to form the overall recognition process. Unless stated otherwise, all symbols appearing in the algorithm have the same meanings as before. A schematic diagram of the algorithm is also depicted in Fig. 4.9 .

Algorithm 4.3

Step 1 Read an input string $s = b_1 b_2 \cdots b_{\ell}$.

Step 2 For $k = 1, W$:

Compute the wM-matrix for string s and the CFG associating with word w_k, using the procedure of section 4.3.2 .

Store $d_m(k) = m(\ell,1,r)$ corresponding to word w_k.

where $d_m(k)$ is the WLD between s and the CFG associated with word w_k.

Step 3 Compute $D_m = \underset{k=1}{\overset{W}{\text{Min}}} \; d_m(k)$

Step 4 Find an index k whose value of $d_m(k)$ equals that of D_m .

If there exist two or more such k indices, go to step 5.

Otherwise, decide word w_k associated with index k to be the required output; END.

Step 5 Let W_N be the number of words associated with the same value of D_m.

For $k = 1, W_N$:

Compute and store $Q(k)$

where $$Q(k) = \sum_{j=1}^{J} \prod_{i=1}^{I(y_k)} p_i(y_k)$$

where y_k is a string generated by the CFG associated with word w_k with a WLD between y_k and s of the value D_m.

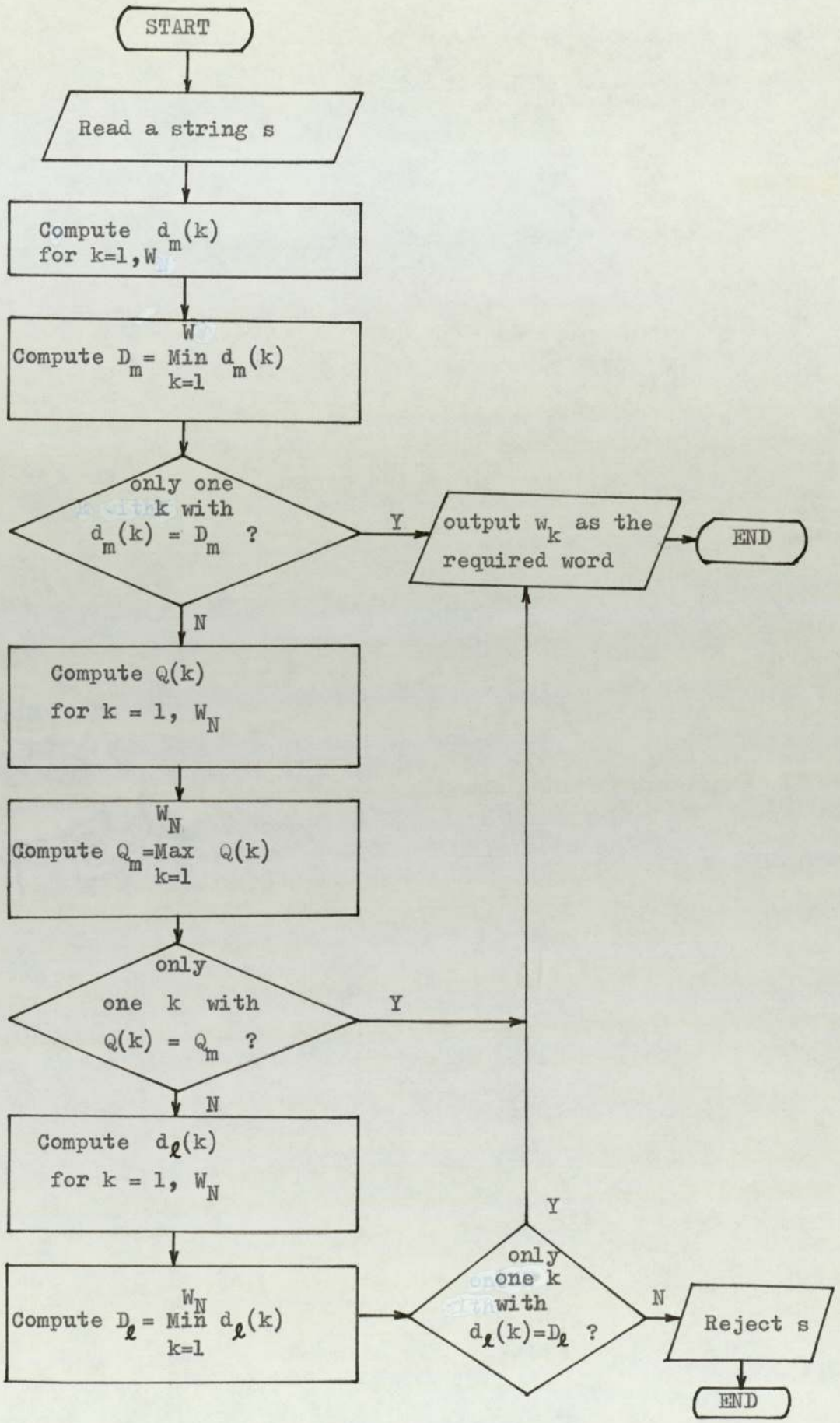


Fig. 4.9 A schematic diagram of algorithm 4.3

$I(y_k)$ is the number of steps in the derivation of y_k.

$p_i(y_k)$ is the probability of the production used at the ith step of the derivation of y_k.

J is the number of distinctively different derivations of y_k.

Step 6 Compute $Q_m = \underset{k=1}{\overset{W_N}{\text{Max}}} Q(k)$

Step 7 If there occur two or more words associated with the same value of Q_m, go to step 8.

Otherwise, decide that w_k is the required word if $Q(k) = Q_m$; END.

Step 8 Compute the WSL of s.

Let W_N be the number of words associated with the same value of Q_m.

For k = 1, W_N :

Compute $d_\ell(k) = \left|\text{WSL of s - AWSL associated with word } w_k\right|$.

Step 9 Compute $D_\ell = \underset{k=1}{\overset{W_N}{\text{Min}}} d_\ell(k)$

Step 10 If there are two or more words associated with the same D_ℓ, reject s and END.

Otherwise, output w_k as the most suitable word, if $d_\ell(k) = D_\ell$; END.

4.7 Discussion

The inference algorithm presented in this chapter employs an incremental method for the construction of nonrecursive CFG's. Consequently, the inferred grammars produce only strings of some finite length. This is appropriate for applications such as automatic recognition of isolated-words, where finite-length strings only are involved.

It is, of course, possible to generate grammars for non-finite languages by modifying the way new nonterminals and rules are appended at each stage of the inference process. One way this can be done is to remove restriction on the recursivity of nonterminals allowed in bi-element rules, for example by permitting productions of the form $A_n \longrightarrow A_m A_n$.

The method presented is guaranteed to generate a proper non-recursive CFG that is capable of producing all the given strings, irrespective of the order in which they are presented to the algorithm. In addition, any other strings created by the grammar will be similar to those in the training set. The method inherently produces compact CFG's having a near-minimal number of rules and nonterminals. These are due to the way the grammar is augmented. At each stage of the process, the algorithm determines the parts of the current grammar that most nearly generate the present string, so that the additions represent minimal change.

CHAPTER 5

MODEL EVALUATION AND EXPERIMENTAL RESULTS

5.1 Basic recognition systems

This chapter is concerned with various evaluation and analytical experiments regarding the application of formal grammars to model a FE in the recognition of isolated words. The experimentation mentioned above involves the use of four basic recognition systems which can be described as follows.

(1) A SFSG-based recognition system using scheme A of the recognition method given in Fig. 3.13

This system will later be referred to as SFS-A recognition system. It is, of course, the system described in chapter 3 and its corresponding flow diagram can be found in Fig. 3.14 .

(2) A SNCFG-based recognition system with the recognition scheme B of Fig. 4.6

Chapter 4 provides detailed descriptions of the system which will be known as SCF-B recognition system. The associated schematic diagram is depicted in Fig. 4.7 .

(3) A SFSG-based recognition system using the recognition scheme B

The above system, hereafter referred to as SFS-B, is implemented and included in the proposed recognition systems for the following reasons. In one of the presentations given below, it is required to evaluate the performances of the inference of two types of grammars, namely the FSG and the CFG, in the modelling of a FE. In such an application, it is necessary for the two recognition systems concerned to use the same recognition scheme. Two options are available for the selection of the required system : either to use the SCF-A or the SFS-B system.

It is found , however, that the former is more difficult to implement than the latter. It is for these reasons that the SFS-B system is implemented. Additionally, this system together with the SFS-A system are also used in comparing the performances of the two recognition schemes A and B . Methods of sections 4.6.2 and 4.6.3 are employed in the recognition part of the SFS-B system. This is illustrated by the flow diagram of Fig. 5.1. The CFG's involved in the method of section 4.6.2 are , of course, replaced by appropriate FSG's.

(4) A recognition system based on direct storage of strings in the training set and using the recognition scheme B

This system whose flow diagram is depicted in Fig. 5.2 will subsequently be known as the stochastic template matching-B (STM-B) recognition system. It is implemented in an attempt to determine whether the use of formal grammars offers any advantage over the direct storage of strings in the recognition of isolated words. In the learning mode of this system, the probabilities of the representative templates are estimated by counting the frequency of occurrence of strings in the training set. The matching of an incoming string to a set of templates during the recognition mode involves the application of the WMN technique described in section 3.4.3 . In this case, the required FTN is directly constructed from a given set of sample strings such that it represents exactly those strings in the training set and no other strings. The principles of sections 4.6.2 and 4.6.3 are again applied as appropriate in determining the decision of the output.

For simplicity and for fast development of the computer programmes involved, all four systems described above are implemented in FORTRAN on a 28K PDP 11v03 minicomputer. Data required for the experiments is taken from a vocabulary of ten digits 'ZERO' to 'NINE' uttered by a single speaker (A.J.PUTMAN) . Putman also designed and

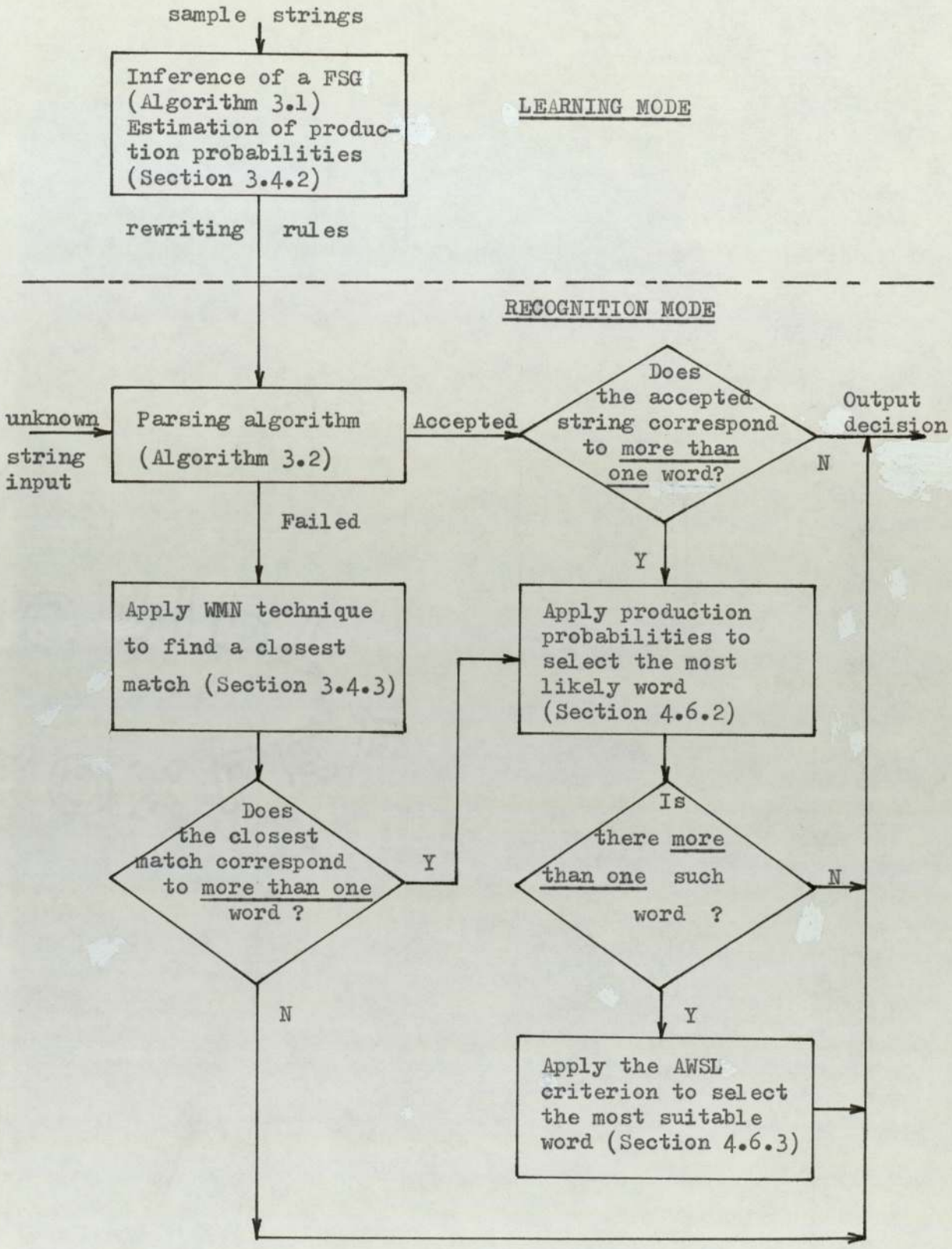


Fig. 5.1 A SFSG-based recognition system using the recognition scheme B

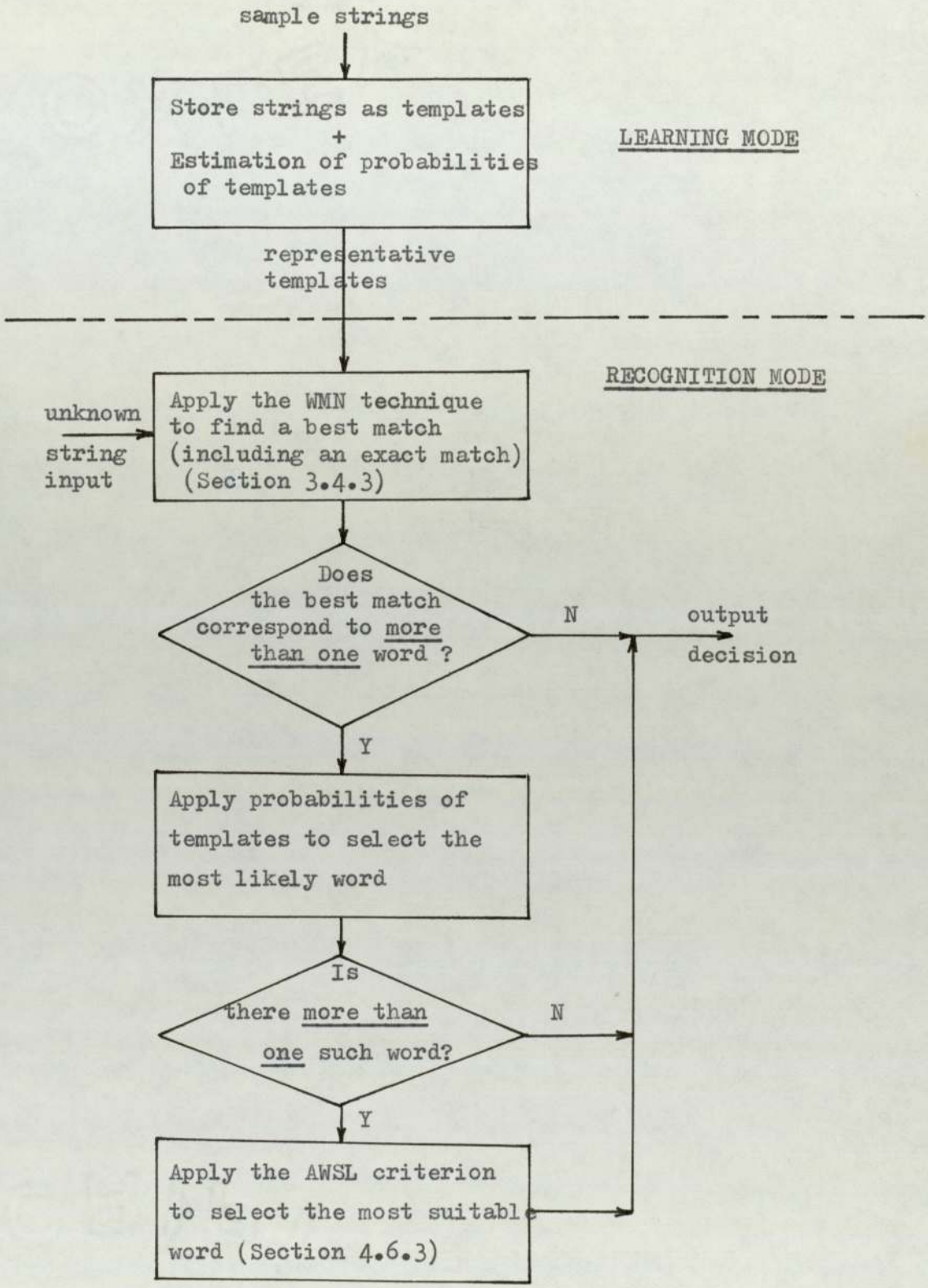


Fig.5.2 STM-B recognition system using template matching and recognition scheme B

built the FE[96] which is used to generate the required data. The speech signal of the spoken digits is of telephone-grade quality. This is obtained from a normal telephone set via a circuit representing two limiting local lines. Deatils of this circuit together with those of the hardware and software parts of the FE can be found in reference 96.

Appendix C gives a training set of 100 strings representing ten spoken digits, each of ten repetitions, obtained from the FE as described above. The symbol strings which will be used as a recognition set are provided by appendix D. The set consists of 500 strings in total with 50 strings for each of the ten digits spoken. Numerical values representing the significance of various symbols can be found in appendix B.

5.2 Evaluation and comparison of models

In this section, some aspects of the modelling of a FE by FSG's and CFG's in the recognition of isolated words are investigated and their results are evaluated. This involves running the appropriate recognition systems described in the previous section using relevant control parameters. The results obtained are then analysed and comparison is made between different models concerned.

5.2.1 Recognition performance

A simple and useful method for evaluating different types of recognition systems implemented in the previous section is to determine their respective recognition performances using the same set of data. The confusion matrices resulting from the test runs of SFS-A, SFS-B, the weighted and nonweighted SCF-B and STM-B systems are given in tables 5.1 to 5.5 respectively. Details of data employed in all test

I \ 0	1	2	3	4	5	6	7	8	9	0	Rej.	Cor. Rec.
1	41		2						7			41
2		43	1				5		1			43
3		8	23	1			1		11	5	1	23
4	1	4		38			3			4		38
5					47					3		47
6						42		8				42
7		7		4			34	1	1	3		34
8				1		12	1	36				36
9	9	1		5			3		27	5		27
0		2	6	3			1	1	3	34		34

Table 5.1 Confusion matrix of SFS-A system ($P_D/P_I/P_S$ = 1.0/2.0/2.0)

I \ 0	1	2	3	4	5	6	7	8	9	0	Rej.	Cor. Rec.
1	41		1				1		7			41
2		44	1				4		1			44
3		8	26	1			1	1	9	4		26
4	1	3		39			3			4		39
5					47					3		47
6						41		9				41
7		7		5			32	2	1	3		32
8				1		11	1	37				37
9	9	1	1	4			2		28	5		28
0		2	7	3			1	1	4	32		32

Table 5.2 Confusion matrix of SFS-B system

I \ O	1	2	3	4	5	6	7	8	9	0	Rej.	Cor. Rec.
1	41		1				1		7			41
2		44	1				4		1			44
3		8	27	1			1		9	4		27
4	1	3		39			3			4		39
5					47					3		47
6						43		7				43
7		7		6			30	2	1	4		30
8				1		9	1	39				39
9	9	1	1	5			2		27	5		27
0		2	9	3				3	4	29		29

Table 5.3 Confusion matrix of SCF-B system (weighted)

I \ O	1	2	3	4	5	6	7	8	9	0	Rej.	Cor. Rec.
1	41		1				1		7			41
2		44	1				4		1			44
3		8	27	1			1		9	4		27
4	1	3		39			3			4		39
5					47					3		47
6						45		5				45
7		7		6			29	2	1	5		29
8				1		11	1	37				37
9	9	1	1	5			2		27	5		27
0		2	9	3			1	3	4	28		28

Table 5.4 Confusion matrix of SCF-B system (nonweighted)

I \ O	1	2	3	4	5	6	7	8	9	0	Rej.	Cor. Rec.
1	41		1				1		7			41
2		44	1				4		1			44
3		8	27	1			1		9	4		27
4	1	3		39			3			4		39
5					47					3		47
6						44		6				44
7		7		6			30	2	1	4		30
8				1		11	1	37				37
9	9	1	1	5			2		27	5		27
0	2	2	9	3				3	4	29		29

Table 5.5 Confusion matrix of STM-B system

system	SFS-A	SFS-B	SCF-B		STM-B
			weighted	nonweighted	
Total Cor.Rec.	365	367	366	364	365
%	73.0	73.4	73.2	72.8	73.0

Table 5.6 Recognition performances of various recognition systems

runs have already been described in section 5.1 . Table 5.6 sums up the overall performances of various recognition systems whose confusion matrices appear in tables 5.1 to 5.5 . To facilitate the presentation and enhance its format, results of the STM-B system are also included in the tables mentioned above, though they will not be discussed until in section 5.3 .

Before the comparison of the recognition performances between FSG and CFG models can be presented, it is essential to discuss some possible sources of errors that cause incoming strings to be incorrectly recognized. Basically, the performance in terms of strings correctly decoded by a recognition system depends on the followings :-

(a) Feature extractor

One significant factor which governs the recognition performance is the degree of overlapping between strings associated with different words. It is possible for a FE to produce very similar strings or even exactly the same strings representing various words in a given vocabulary. For example, string 'ℓ' appears in both of the words 'TWO' and 'THREE' of the testing set given in appendix D. In another illustration, string 'JoC' from word 'FOUR' of the training set in appendix C also appears in word 'SEVEN' of the testing set. For the occurrence of such strings,it immediately follows that the strings concerned will be misrecognized. Thus, the importance of a FE and its influence on the overall performance of an IWR system and the need for a good FE cannot be overemphasized. In general, the basic criterion governing the design of a FE is to obtain the largest possible intra-string distances (LD or WLD) between all words concerned. This should cut down the number of overlapping strings with the consequent improvement of the recognition performance.

(b) Learning algorithm

The learning part of a grammar-based recognition system can contribute to the overlapping of strings and hence inducing errors in the recognition process in the following manner. The grammar inferred for one word can predict or generate strings, in addition to strings in the training set, which are similar or very similar to strings of other words. This depends to a large extent on the intra-distances of strings between different words in the training set and to some extent on the learning algorithm used. As an example, string 'BdEj@h' in word 'SIX' of the testing set is wrongly recognized as word 'EIGHT' by the weighted SCF-B recognition system . This is because the inferred grammar associated with word 'EIGHT' predicts an additional string 'Gj@j' from strings 'Gj@Bf' and 'GjHj' in the training set of the same word. This in turns is caused by a small WLD between strings 'Gj@Bf' and 'Gk@f' in words 'EIGHT' and 'SIX' respectively in the training set. Since the string under test 'BdEj@h' resembles more closely (in terms of WLD) to the predicted string 'Gj@j' than to the training string 'Gk@f', the recognition system gives the incorrect output decision as described earlier.

In an effort to restrain the occurrence of the above situation as far as possible, many learning algorithms are formulated on the basis of the following requirement. The algorithm should be such that the inferred grammar, apart from producing strings which are similar to the ones in the training set of the same word, generates as small as possible the number of strings that are closely resembled to strings of other words. One criterion usually adopted to satisfy the above requirement is to construct the inference algorithm so as to maximize the similarities between strings corresponding to the same word in the training set. As described in chapters 3 and 4, all learning algorithms

presented in this thesis employ the above criterion in the construction of various grammars.

(c) Probabilistic part of recognition algorithm

In the recognition process, it frequently happens that two or more grammars could have equally generated an incoming string with the same minimum penalty (LD or WLD) incurred. This nondeterministic situation resulting from either or both of the sources in (a) and (b) can be broadly divided into two groups. In the first category, one of the candidate grammars correctly produces the string whilst in the second group none of the grammars provide the correct recognition. Methods using probabilities have been developed in both recognition schemes A and B to select only one grammar which is the most suitable according to some criteria. For obvious reasons, the second group of the nondeterministic situation inevitably yields incorrect decision irrespective of whatever probabilistic method is used. For the first group, the methods can make a wrong decision which may be caused by the inadequacy of the sample set used in the estimation of production probabilities.

As an example, consider the classification of string 'FjC' taken from word 'THREE' of the testing set using the weighted SCF-B system. The string 'FjC' could have equally been derived from grammars of words 'THREE' and 'NINE' with the same minimum penalty of 5 from strings 'EiF' and 'IhC' respectively. Since the string probability of 'IhC' is greater than that of 'EiF', 'NINE' is wrongly selected as the word most likely to correspond to string 'FjC'. Although the above shows some defects of the probabilistic parts of recognition algorithms, the methods still provide better performances when compared with an arbitrary selection of one grammar from a set of equally suitable grammars in the nondeterministic case.

From the inspection of table 5.6, it can be seen that the highest recognition performance that can be achieved is only 73.4% (SFS-B system). This is much lower than the performance normally claimed by many experimental systems for the recognition of isolated-words using the same vocabulary. For example, the system of White[19] is reported to obtain around 96 % correct recognition for a vocabulary of the same ten digits. By investigating further, it is found that most of the errors (about 90 % of total errors) occurring in all recognition systems of table 5.6 are due to the FE as already described in part (a) of sources of errors. If most of these errors were rectified, the overall performances of the recognition systems in table 5.6 would become comparable to that mentioned in the literature. Since, according to reference 96, only about one-third of useful features extracted from the input speech signal are used in the encoding of symbol strings such as those given in appendices C and D, it is hardly surprising that the overall recognition performances stated in table 5.6 do not measure up to those of comparable systems appearing in the literature. It is also interesting to notice that 'THREE' appears to be the worst recognized digit as shown by the given confusion matrices. This may result from the difficulty in pronouncing the digit such that the generated strings do not resemble too closely with strings of other digits.

From tables 5.2,5.3,5.4 and 5.6, it seems that the use of FSG's and CFG's to model a FE offers comparable recognition performances with less than 1 % variation between any of the associated systems. This, in a way, is to be expected since learning algorithms for the inference of both types of grammars are based on a similar criterion of maximizing the similarities between strings in the sample set. Thus, there appears to be no advantage, as far as the recognition performance is concerned, for the use of CFG approach over that of FSG in the modelling of a FE

in isolated-word recognition. The above is true for grammars inferred in this thesis and it is expected to hold true for general FSG's and CFG's provided they are constructed on the basis of similar criteria.

Comparison is now made between the use of weighted and non-weighted M-matrices in the inference of CFG's. Again, there appears to be no significance differences between the weighted and nonweighted versions of the SCF-B system, though the former gives a slightly better performance than the latter. This is because the learning algorithm using the weighted M-matrix is provided with additional information about the training data via the knowledge of significance of various symbols.

Performances of two recognition schemes A and B are considered next. Although tables 5.1, 5.2, and 5.6 show the performance of scheme B to be slightly better than that of scheme A, the differences obtained are not significant enough to suggest the superiority in the recognition performances of B over A. However, since it is easier to implement scheme B than to do scheme A, the former is preferable to the latter. In the recognition scheme A, the values of P_D, P_I, and P_S given in table 5.1 represent the deletion, insertion and substitution coefficients of equations 3.17 to 3.19 respectively. These values are the design parameters and are determined experimentally in an attempt to improve the recognition performance. For the data given in appendices C and D, the improved performance achieved when $P_I = P_S = 2P_D$ compared with $P_D = P_I = P_S = 1$ indicates that insertion and substitution events of the recognition scheme A are equally likely to occur and that both are more likely than the deletion event. These design parameters can be adjusted experimentally to suit a given set of data

5.2.2 Measure of complexity

This section presents the comparison between FSG and CFG models in terms of computational requirements or complexity measure of the grammars. Appendix E gives the rules of FSG's, of the weighted and nonweighted versions of CFG's constructed directly from the training set in appendix C.

In general, the measure of complexity of the required grammars involves the determination of the followings :-

(i) The length of the longest member in any rewriting rule, L_m, of the grammar;

ie. $L_m = \text{Max}\ (|\alpha|, |\beta|)$ for all $\alpha \longrightarrow \beta$ in R (5.1)

Due to the formats of the rules in the grammars concerned, all FSG's and CFG's inferred in this thesis have $L_m = 2$. This is the smallest L_m that can be associated with any grammar apart from grammars which generate only single-symbol strings.

(ii) Number of terminals and nonterminals created by the grammars.

(iii) Number of rules in the grammars.

The number of terminals, nonterminals and rules mentioned in (ii) and (iii) of various inferred FSG's and CFG's are presented in tables 5.7 to 5.9 . Notation of symbols appearing in these tables is as follows:-

$\|\Sigma\|$ = number of terminals in a grammar

$\|V_N\|$ = number of nonterminals in a grammar

$\|V\|$ = number of terminals and nonterminals in a grammar

$\|R\|$ = number of rules (total) in a grammar

$\|R_T\|$ = number of terminating rules in a CFG

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO
$\|\Sigma\|$	13	10	11	19	27	18	18	21	12	15
$\|V_N\|$	9	7	8	20	26	22	22	24	8	26
$\|V\|$	22	17	19	39	53	40	40	45	20	41
$\|R\|$	16	16	15	31	42	40	34	39	16	46

Table 5.7 Complexity measure of inferred FSG's

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO
$\|\Sigma\|$	13	10	11	19	27	18	18	21	12	15
$\|V_N\|$	4	3	3	14	29	26	19	21	3	39
$\|V\|$	17	13	14	23	56	44	37	42	15	54
$\|R_T\|$	13	10	11	19	27	18	18	21	12	15
$\|R_B\|$	3	2	2	13	28	28	18	22	2	38
$\|R_{St}\|$	8	10	8	10	10	10	10	10	9	10
$\|R\|$	24	22	21	42	65	56	46	53	23	63

Table 5.8 Complexity measure of inferred CFG's (weighted)

	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO
$\|N\|$	13	10	11	19	27	18	18	21	12	15
$\|V_N\|$	4	3	3	14	29	28	18	21	3	31
$\|V\|$	17	13	14	33	56	46	36	42	15	46
$\|R_T\|$	13	10	11	19	27	18	18	21	12	15
$\|R_B\|$	3	2	2	13	28	29	17	22	2	34
$\|R_{St}\|$	8	10	8	10	10	10	10	10	9	10
$\|R\|$	24	22	21	42	65	57	45	53	23	59

Table 5.9 Complexity measure of inferred CFG's(nonweighted)

system / hr:min:sec	SFS-A	SFS-B	SCF-B		STM-B
			weighted	nonweighted	
Training time	0:0:18	0:0:18	0:1:13	0:1:00	0:0:18
Recognition time	0:26:25	0:25:12	1:20:18	1:19:25	0:22:11

Number of training strings = 100

Average length of training strings = 3.56 symbols/string

Number of testing strings = 500

Average length of testing strings = 3.45 symbols/string

Table 5.10 Time required for training and testing of various recognition systems

$\|R_B\|$ = number of bielement rules (not including start rules) in a CFG

$\|R_{St}\|$ = number of start rules in a CFG

(iv) Complexity of learning and recognition algorithms.

(a) for FSG's

The number of operations for the construction of rules of a FSG from a given string is roughly proportional to the string length. In the recognition process, the size of the current WMN governs the complexity required to classify an incoming string. This depends on the number of rules in the grammar concerned and the length of the string and can be formally expressed as follows.

Number of operations required to classify a string of length ℓ using $\|R\|$ rules in a given grammar = $(\ell + 1).\|R\|$ (5.2)

As in the learning process, the number of operations required in the recognition of a string is again proportional to the length of that string.

(b) for CFG's

Since the M-matrix (either weighted or nonweighted) is employed in both the learning and recognition processes, the required complexity for CFG's is obtained by determining the number of steps necessary in the computation of the M-matrix for a string of length ℓ. This can be estimated as follows.

Terminating rules

for i = 1 and from equation 4.11 :

no. of steps required = $\|R_T\| \cdot \ell$ (5.3)

for i > 1 and from equation 4.12 :

no. of steps required = $\|R_T\| . \sum_{i=2}^{\ell} (\ell+1-i).(i-1)$ (5.4)

Bielement rules

for i = 1 and from equation 4.13 :

$$\text{no. of steps required} = \left(\|R_B\| + \|R_{St}\| \right) . \ell \qquad (5.5)$$

for i > 1 and from equation 4.14 :

$$\text{no. of steps required} = \left(\|R_B\| + \|R_{St}\| \right) . \sum_{i=2}^{\ell} (\ell+1-i)(i-1) \qquad (5.6)$$

combining equations 5.3 to 5.6 yields,

total number of steps required

$$= \left(\|R_T\| + \|R_B\| + \|R_{St}\| \right) \ell + \left(\|R_T\| + \|R_B\| + \|R_{St}\| \right) \sum_{i=2}^{\ell} (\ell+1-i)(i-1)$$

$$= \|R\| \left\{ \ell + \sum_{i=2}^{\ell} \left[\ell - (i-1) \right] (i-1) \right\}$$

$$= \|R\| \left\{ \ell + \sum_{i=2}^{\ell} \ell(i-1) - \sum_{i=2}^{\ell} (i-1)^2 \right\}$$

$$= \|R\| \left\{ \ell + \frac{1}{6} \ell(\ell^2 - 1) \right\}$$

$$= \frac{\|R\|}{6} (\ell^3 + 5\ell) \qquad (5.7)$$

That is, the number of operations required either to construct rules of a CFG from a string or to decode an unknown string is proportional to the cube of the length of the string concerned. This is in accordance with the complexity expected of a CFG. Table 5.10 presents the complexity of the learning and recognition algorithms in the form of the time required for training and testing of strings for various recognition systems. The table also includes the results associated with the STM-B system for the same reason given in the previous section.

Generally, the use of CFG's should provide models that are more compact than those obtained from the approach of using FSG's. That is, the number of rules and nonterminals in the former case should be smaller than those in the latter. However, results in tables 5.7 to 5.9 indicate that this is not so. Two main factors account for the

above situation which is not unexpected. First, it is caused by the inherent characteristics of the Chomsky normal-form grammars employed in the system. A normal form CFG, due to its format, usually requires a larger number of nonterminals and rules than a FSG does in the representation of the same set of strings. Secondly, the way the learning algorithm is formulated for CFG's also contributes to the increase in the number of the corresponding rules and nonterminals in the following manner. The algorithm applies a constraint on the extent which the number of nonterminals and/or rules can be reduced by the possible merging of similar segments of various strings. This is done to ensure that the inferred grammar does not generate too many strings which are similar to strings of other words.

From equations 5.2 and 5.7 and table 5.10, it can be seen that the CFG approach requires a larger amount of computation in both the training and recognition operations than the amount involved in the approach of using FSG's in the modelling of a FE. This, in a way, is to be expected since the increased descriptive power of strings obtained from the use of a more general class of grammars has to be paid for in terms of the increase in the computation required of the system.

The foregoing presentation, thus seems to indicate that there is no advantage gained in terms of computational requirements of the systems concerned for the use of CFG's over that of FSG's for the modelling of a FE in isolated word recognition.

5.2.3 Discussion

From the results of sections 5.2.1 and 5.2.2, it appears that there is no advantage for the use of CFG's over the FSG approach, as

far as recognition performance and computational requirement are concerned, in isolated word recognition. This may be because, as the name implies, only a single isolated word, and not a complete sentence, needs to be recognized and this does not require the knowledge of the syntax of the word concerned.

There are, however, other situations where there may be an advantage for CFG approach. One of these is the recognition of connected or continuous speech. One difficult problem in continuous speech recognition is the determination of word and sentence boundaries, which, unlike the case in isolated speech where words are spoken in isolation, are usually obscured. Another problem is that acoustic parameters of words pronounced connectedly are, depending on the context, very different from those obtained from the same words spoken in isolation.

The characteristics of continuous speech as described above can induce errors in various words spoken. Other possible sources of errors include the inadequacies of many processes in the earlier stages of the system such as segmentation and transcription of acoustic data, the introduction of spurious words and the presence of foreign noises. In such a situation, it is desirable to be able to start processing at any point in the sentence in an attempt to uniquely identify a correct or least-error word. Once a starting point representing a correct word has been pinpointed, other words or phrases can then be predicted by the syntax recognizer on the basis of the inferred grammar and local context. The above requires a parser which is capable not only of proceeding from left to right or vice versa but also of starting anywhere in the utterance and continuing to parse in both directions.

The use of a parser based on a FSG thus seems to fall short of the above requirements. On the other hand, a CFG, because of its greater generative power in the sense that the grammar does not require to produce terminal symbols in a strictly left-to-right order, can be used in the above situation. Thus, the application of CFG's in continuous speech can provide some sort of advantage such as that already described, when compared with the FSG approach. Other applications where the CFG approach may prove useful in the description of the language concerned include the analysis of chromosome, picture and scene analysis, character recognition, recognition of two-dimensional mathematical expressions and finger print identification[45,51-53,97].

5.3 Symbol-source modelling versus direct storage of strings

This section investigates the pros and cons between the approach of using formal grammars to model a FE by constructing rules from sample strings generated from the FE and that of directly storing the strings ie. template matching approach in isolated word recognition. In the comparison of the two approaches, it is necessary to apply the same recognition scheme to various recognition systems concerned. This is done to ensure that conclusions drawn from the comparison tests are independent of the recognition scheme used and only depend on the method of representing sample strings. This is because the use of different recognition schemes in the approaches can affect the final outcome of the comparison in such a way that the result obtained is incorrect and misleading. It is decided to select scheme B, for reasons given in section 5.1, as the required recognition scheme. Thus, the recognition systems concerned are SFS-B, SCF-B and STM-B.

The advantages and disadvantages associated with the two approaches can be described as follows.

(i) One of the advantages of representing strings as a set of rules of formal grammars instead of simply storing the strings themselves is that when the language is very large or even infinite, it would be impractical or even impossible to store the strings. In addition, it is neither desirable nor possible to put an upper limit on the length of the longest strings in the languages of many applications such as chromosome analysis[45] or finger print identification problems[97]. This type of languages cannot be specified by an exhaustive enumeration of the strings of the language concerned. Thus, the representation of strings by means of formal grammars provides a capability for using a set of rules of finite size to describe a set of strings which may not be finite. An attractive aspect of this capability is the use of the recursive nature of a grammar as illustrated by the following example. A non-finite language consists of strings ab^nc for n = 1, 2, ... can be represented by a grammar whose rules are :-

£ ⟶ aA B ⟶ bB

A ⟶ bB B ⟶ c

where £, A, B are nonterminals and a, b, c are terminals.

(ii) For many problems of pattern recognition, not only the classification of patterns but also their descriptions are required in the determination of the solution. Such problems include chromosome analysis[45], picture processing and scene analysis[51], character recognition[52], recognition of two-dimensional mathematical expressions[53], finger print identification[97] and continuous speech recognition[98]. For these applications, methods based only on the classification mechanism such as the template matching technique may, by themselves, be inadequate. It is then necessary to employ syntactic methods such as the formal-grammar approach to explicitly exploit the

structural relations of the patterns in the description process.

(iii) The use of symbol-source models makes possible the 'generalisation' of strings in the training set. In other words, in addition to the training strings, the inferred grammar also predicts or generates other strings which are similar to the ones in the training set. This means that the formal grammar approach includes a wider range of strings than does the approach of using template matching technique for a given training set. Thus, a larger sample size is needed if the latter is to cover the same number of strings as for the case of the former. For example, the CFG inferred in section 4.5 requires only 6 training strings in order to cover strings s_1 to s_7 whereas if the method of direct storage of strings is used, it will be required to store 7 strings to achieve the same result. Incidently, the CFG also predicts three additional strings which are similar to s_1, s_2, and s_6 respectively. Thus, the inferred CFG covers a total of 10 strings from a sample set of 6 strings. In another example, string 'HoC' from the recognition set in appendix D is correctly predicted as word 'FOUR' by the FSG inferred in chapter 3.

Table 5.11 displays strings in the testing set that are correctly recognized as a result of 'generalisation' created by the use of formal grammars.

Although, the formal grammar approach does give correct recognition to many strings, as shown in table 5.11, which are misrecognized by the approach of using template matching technique, the overall recognition performance of the former is only slightly better than that of the latter as illustrated by table 5.6 . This, unfortunately, is caused by factors described in section 5.2.1 .

Grammars	FSG				CFG(weighted)
Digits	SIX	SEVEN	EIGHT	ZERO	EIGHT
Strings	cFℓ@BgC Gi@Ce	EdDi	Bc@Bd Bg@a Hh@f	CfDc ChFf BeCf GiGf	Fi@h Dk@Bj

Table 5.11 Strings correctly recognized due to the use of formal grammars

By the inspection of various rules given in appendix E, other strings generated by the grammars concerned in addition to strings in both the training and recognition sets can also be determined.

(iv) For the approach of using FSG's, it is possible to obtain some reduction in computation involved in the recognition process when compared with the method of direct storage of strings. As demonstrated by table 5.10, the time required for training a given set of strings is the same for both cases mentioned above. The CFG approach is not considered here since it requires, for reasons given in section 5.2.2, a larger amount of computation than do the methods of FSG's and template matching.

The reduction in computation obtained from the FSG approach is made possible because of the use of merging between various segments of similar strings during the training process. This leads to a reduction in the number of nonterminals and rules produced by the grammar. This, in turns, cuts down the computational requirements of the recognition process.

The following presents the estimation of number of operations required in the computation of the WMN's which govern the necessary amount of computation of the recognition algorithms in the FSG and the template matching approaches.

The number of operations needed to compute a WMN constructed from a FTN of Z links and an incoming string of length ℓ is equal to :

$$(\ell + 1) \cdot Z \qquad (5.8)$$

For a FTN created from a FSG, the value of Z is equal to the number of all rules of the grammar, as given by equation 5.2 .

ie. $$Z = Z_F = \|R\| \qquad (5.9)$$

where Z_F is the number of links of a FTN associated with the FSG approach.

In the approach of direct storage of strings, each and every distinct string in the training set is compared with an incoming string to obtain the best match. This requires a set of WMN's each of which is constructed from each distinct training string.

Thus, $$Z = Z_T = \sum_{i=1}^{M_D} \ell_i \qquad (5.10)$$

where Z_T = number of links of a set of FTN's associated with template matching approach

ℓ_i = length of the ith distinct string in the training set

M_D = number of distinct strings in the training set .

Table 5.12 presents values of Z_F and Z_T for the training strings of appendix C. The values of Z_F are, of course, the corresponding values of $\|R\|$ in table 5.7 .

Digits	ONE	TWO	THREE	FOUR	FIVE	SIX	SEVEN	EIGHT	NINE	ZERO
Z_F	16	16	15	31	42	40	34	39	16	46
Z_T	19	17	15	33	48	52	39	44	20	58

Table 5.12 Number of links of FTN's for FSG and template matching approaches

From the results in table 5.12, it can be seen that the approach of using FSG to describe a given set of strings provides a reduction in the computation of the WMN compared with the approach of direct storage of strings. This, however, is not reflected in the recognition time of the SFS-B and STM-B systems given in table 5.10. This is because the rules of the inferred FSG's have not been rearranged in the ascending order of the LS nonterminals as those given in appendix E. That is, rules are created and stored in the system memory according to the order of presentation of strings. Consequently, it is necessary to compute the WMN at least twice for the FSG approach to ensure that contents of all elements of the WMN reach steady-state values. The foregoing computation can be speeded up if the rules are arranged, after the training operation, in the proper sequence as described above.

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

6.1 Future work

6.1.1 Real-time problem

Many implementations of word recognition algorithms are carried out using a computer. This may not be fast enough for some practical applications where real time responses are required. This section suggests the possible use of special purpose hardware in addition to a general purpose computer in an attempt to implement the algorithms in real-time. Only parts of algorithms that require large amount of computation will be considered. This implies the implementation of the computation of the WMN in FSG models and that of the M-matrix in the CFG approach.

(a) Implementation of the computation of the WMN

Fig. 6.1 outlines a possible scheme of implementing the computation of the WMN using special hardware. Each rectangular unit represents hardware implementation of a FTN and consists of as many storage elements as the number of nodes in the FTN. All storage elements are interconnected according to the configuration of the FTN and each connecting link is associated with only one symbol determined by the grammar concerned. There is also a special logic circuit for each storage element whose incoming links form the input of the circuit. The function of the logic circuit is to select the largest of the values presented by incoming links for a given input symbol. These rectangular units are repeated as often as the length of the string to be analysed and are connected in the same way as the corresponding WMN. The logic circuits mentioned in Fig. 6.1 receive a short pulse from the clock every time an input symbol appears. Outputs of these circuits are such that only the rectangular unit which corresponds to the current

symbol is activated. For example, unit 2 is associated with the 1st symbol and unit 3 with the 2nd symbol and so on. The circuits also provide two pulses for each pulse from the clock. The first pulse is used to control the computation of various elements in the present unit due to elements in the preceding unit. The second activates the computation within the same unit. Output from the appropriate element of the final unit associated with the last symbol gives the WLD as required.

The processor described above is for one grammar only. To increase the speed of operation, similar processors corresponding to other grammars need to be applied in parallel. The grammar with the smallest value of WLD is then selected as the grammar which could have generated the string.

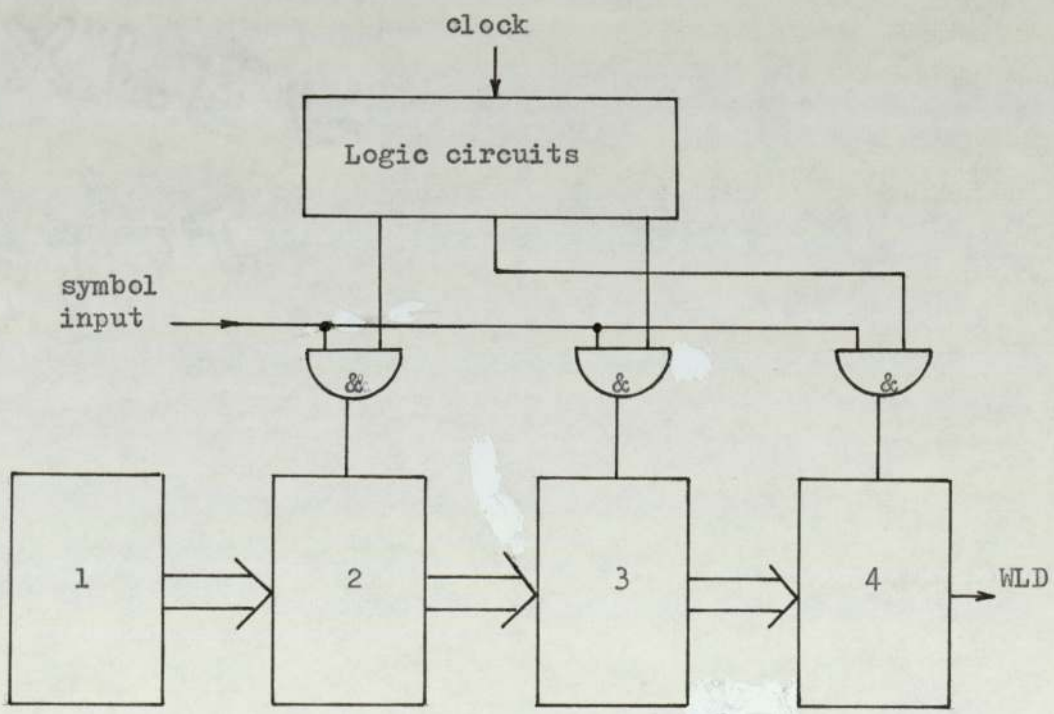


Fig. 6.1 A hardware scheme for the WMN and a length-3 string

(b) Implementation of the computation of the M-matrix

A generalised block diagram of a hardware scheme for the computation of the M-matrix and a string of length 3 symbols is depicted in Fig. 6.2 . Elements m_{ijk} of the M-matrix together with appropriate logic circuits are represented by rectangular units as shown. There are as many storage elements in each unit as the number of distinct LS nonterminals in the corresponding types of rules ie. terminating rules or bielement rules. A special unit contains values of hierarchy levels of all nonterminals predetermined from the rules of the CFG. Input symbols are presented to units corresponding to terminating rules and i = 1 under the control of logic circuits and clock pulses in the same way as in the case of WMN. Elements of these units are then computed. Other units are computed in the order shown in Fig. 6.2 . That is, the remaining elements corresponding to terminating rules and elements associated with bielement rules for i = 1 are processed at the same time. Other units for bielement rules are then computed in the order of increasing values of i, the substring length, until the WLD is found. As in case (a), the above processor is repeated for other grammars and they are applied in parallel in order to speed up the computation involved.

The above schemes of computing WLD involves the processing of all strings in the dictionary ie. all strings generated by the grammars concerned. This may take too long if the dictionary is large. In order to reduce the time required, it may be necessary to use some other methods in addition to the above schemes eg. techniques involving n-grams might be used.

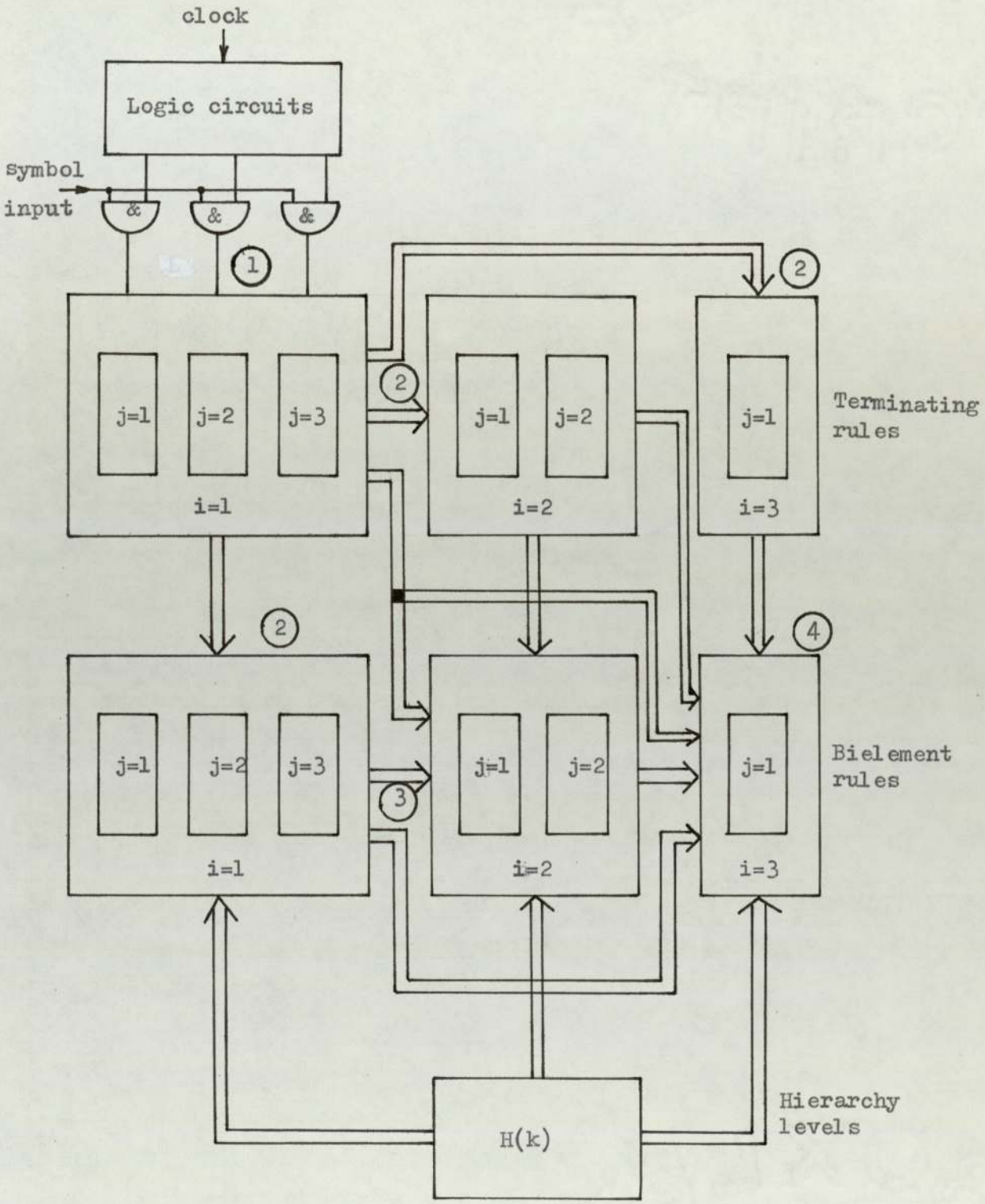


Fig. 6.2 A hardware scheme for the M-matrix and a length-3 string

6.1.2 Improvements of recognition performance

For various recognition systems implemented in this thesis, many of the errors in word recognition are due to the characteristics of the FE used as described in section 5.2.1 . The most obvious way to improve word accuracies is to develop a better and more sophisticated FE. However, this approach, which is still a research problem, is outside the scope of the work reported in this thesis and consequently will not be investigated here. Other possible methods that can improve the recognition performance are as follows :-

(i) As mentioned earlier, the overlapping between strings of different words is mainly caused by the fact that the transcribed symbol strings represent only a small fraction of acoustic parameters extracted from the FE. The method is then to find a way of utilizing a large number of parameters in the construction of the models. One such approach is to use several symbols simultaneously instead of only one symbol at a time, where each symbol represents a different parameter. This is the concept of 'vector valued features'. One simple solution is to build a grammar for each of the parameters extracted for one word. In the recognition mode, each of the strings representing various parameters associated with the word spoken is individually processed by the appropriate grammar. The final decision as to which word has been spoken is determined, say, by the majority votes of the strings concerned.

(ii) It is commonly appreciated that as well as the order of appearance of symbols in a string, but also their duration are important in the descriptions of a spoken word. Consequently, concept of duration of features can be used to improve the performance of an inferred grammar. This can be implemented as self-loops in the FTN's or PTN's with probabilities of the loops denoting duration lenghts. In the case of

FTN, this is similar to the quasi-Markov process.

(iii) The use of the negative sample set, if available, in addition to the positive one, can improve the system's performance. That is, if a set of strings is known not to belong to a given word, the grammar corresponding to that word can then be modified such that these strings are excluded from the language of the grammar. The problem in this case is to find such a negative sample set for a particular application.

(iv) In many pattern recognition problems, the frequencies of occurrence of different types of errors, namely insertion, deletion and substitution, depend on the nature of the application concerned. For example, optical character recognition rarely introduces insertion or deletion errors. By observing such characteristics of a given data set and applying this knowledge in the recognition algorithms, it becomes possible to obtain an improvement in the recognition performance. One way to achieve this is by setting appropriate multiplicative factors (instead of unity) in equations 3.11, 3.13, and 3.14 for the case of FSG approach and equations 4.15, 4.17, and 4.18 for the CFG approach. The foregoing is for WLD's. For applications involving LD's, the corresponding equations that need to be modified are 3.16, 4.9, 4.11, and 4.12 .

(v) Another approach that may improve the recognition performance involves the introduction of various restrictions to the formats and the applications of productions. This may, for example, include labelling the productions and coding of productions in terms of level numbers according to the hierarchical significance of the productions. This method of imposing restrictions on the productions may be well suited to the situation where there are many overlapping strings.

6.1.3 Other work

One area of interest that is worthy of further investigation is the study of the effects of various telephone impairments on the models constructed in this thesis. This involves experimental tests of appropriate recognition systems under the insertion of controlled degradations such as continuous noise, variable frequency characteristics and nonlinear distortion.

Another area is to extend this work to cover the recognition of connected speech. In particular, it will be of interest to confirm, or otherwise, the inadequacy of the use of a FSG in continuous speech and also of the advantage of the CFG approach over that of the FSG as suggested in section 5.2.3 . It is also of interest to study the practicality of the application of formal grammars to the synthesis of speech which is the reverse process of this work.

6.2 Conclusions

An automatic isolated-word recognition system normally consists of a feature extractor or a preprocessor of some sort followed by a recognizer or a recognition processor. Because of the inherent variations in speech when a word is uttered even by the same speaker, it is necessary to incorporate some form of 'training' or 'learning' process into the system.

Apart from the classical decision-theoretic methods, techniques of formal language theory or the syntactic methods provide another useful approach to the solution of classification and description in a speech recognition system. The linguistic method proves to be very attractive to use due to the availability of mathematical linguistics

as a tool. The method also seems to be well-suited to the problem of an IWR system where only a finite number of features are generated for each utterance.

The application of linguistic approach to an IWR system can be viewed as the process whereby formal grammars are employed to model the FE whose characteristics are very little, if at all known.Basically, the method works as follows. In the training stage, sets of syntactic rules or grammars are constructed, one for each word in the vocabulary, directly from a given set of sample strings of features represented by symbols. Constructive approach of grammar inference is chosen so that model of the FE can be formed more realistically. Supervised learning is also assumed. In the recognition mode, an incoming string is analysed to determine which grammar, if any, could have generated it. The word corresponding to such grammar is then said to have been recognized.

In IWR systems, unlike many applications of grammar inference where the class of grammars to be inferred is precisely defined, it is not clear what types of grammars best represent the FE. Only two types of grammars are considered here, namely the FSG's and the CFG's. The FSG approach is selected initially because of its simple and well-established characteristics and its sequential nature similar to that of the string symbols. In addition, many efficient computational techniques are known for the FSG methods. The CFG approach is introduced in an attempt to determine whether there is any advantage from the use of a more powerful grammar in isolated-word recognition.

Inference algorithms of both approaches are based on the criterion of maximizing the similarity between various strings of the same word. The basis of the inference process which applies to both FSG's and CFG's can be explained as follows. The skeleton grammar G_1 is

first constructed from the first string in the sample set such that G_1 can generate only that string. Other strings are then individually processed in the search for incompatibility between each string and the current grammar. If the nth observed string s_n can be derived from the (n-1)th inferred grammar G_{n-1}, then $G_n = G_{n-1}$ and no augmentation of G_{n-1} is required. Otherwise, G_{n-1} is augmented such that G_n is produced which can generate the present string. For the CFG approach, the matching process between an incoming string and an existing CFG requires the computation of the minimisation matrix, M, whose elements reveal the compatibility or otherwise between the former and the latter.

There are two different recognition schemes, A and B, employed in various recognition processors. In scheme A, an incoming string is tested to determine whether there exists an exact match for the string. In the case of unsuccessful matching, an attempt is made to find a closest match for the string. This is supplemented, if necessary, by a stochastic algorithm to select only one word that is the most likely to correspond to the string. For the case where the exact match is associated with two or more grammars, another stochastic algorithm is applied to select only the most likely grammar. For the recognition scheme B, an attempt is made to find a best match which also includes an exact match for an incoming string. A stochastic technique is applied if two or more grammars are equally likely to have generated the string. If, after applying this technique, the output is still undecided, a selection is made of the most suitable word according to the AWSL criterion.

The recognition algorithms of both schemes are thus not too restrictive in the sense of immediate rejection of an erroneous string but rather trying to find a grammar that could most likely have generated the string. This can be very useful in many applications involving

noisy strings. In the FSG approach, the WMN technique based on the principle of dynamic programming is employed to find the best match for an incoming string. The determination of the best match for a string in the CFG approach is accomplished by using the wM-matrix (or M-matrix) as a recognition matrix. Performances in terms of number of strings correctly recognized of the two recognition schemes are comparable with one another. However, scheme B is preferable to scheme A because it is easier to implement the former than to do the latter.

Both the FSG and CFG models offer comparable recognition performances with less than 1 % variation in word accuracies between any of the associated systems. The increased descriptive power of strings obtained from the use of a more powerful CFG is, as expected, paid for by the increase in the amount of computation required of the system concerned. Consequently, there appears to be no advantage gained in terms of recognition performance and computational requirement, from the use of CFG approach over that of FSG in the modelling of a FE in isolated-word recognition. This may be because the isolated-word application does not require the knowledge of the syntax of the word to be recognized since only a single isolated-word, and not a complete sentence, is required to be recognized.

The representation of strings by a set of rules of formal grammars instead of direct storage of strings makes possible the 'generalisation' of strings in the training set. That is, the inferred grammar generates, in addition to the training strings, other strings which are similar to the ones in the training set. This means that a larger sample size is needed for the approach of using template matching technique if it is to cover the same number of strings as for the case of formal grammar approach. The approach of using FSG's

also provide a reduction in the amount of computation required by the recognition process. This is possible because of the reduction in number of nonterminals and rules produced by the grammar as a result of the merging between similar segments of strings during the training process.

The use of 'linguistic' variables instead of or in addition to numeric variables provides an effective and useful means of approximations of complex or ill-defined systems such as the FE, where it is difficult or even impossible to apply precise mathematical analysis. Experimentation, though expensive in terms of labour and equipment, is essential to automatic speech recognition problem as to many other applications in pattern recognition such as chromosome analysis in biomedical application. This is because it is difficult to predict the required recognition performance theoretically due to noisy nature of strings involved. It is hoped that the knowledge and experience gained from the design, construction and experimentation of speech recognition systems will pave the way to better and increased insight into speech perception in humans.

APPENDICES

APPENDIX A

A METHOD FOR TESTING THE RECURSIVENESS OF A FSG

The following presents a method for testing whether a given FSG is recursive or not. It is based on the construction of a transition matrix T whose elements represent the number of direct paths between different states of the associated FTN. The method can be simply explained as follows. The transition matrix T is repeatedly being multiplied by itself until for some value of n either all elements of T^n are zero OR two or more diagonal elements of T^n are no longer zero. The grammar is then said to be nonrecursive in the former case and recursive in the latter. It is assumed that the grammar does not contain rules of the form $A \longrightarrow aA$ where 'A' and 'a' are a nonterminal and a terminal respectively. In other words, there are no self loops in the corresponding FTN. A method is also given for computing the number of distinct strings whose lengths do not exceed n that can be derived from the grammar.

Formally, the method can be described in the following steps.

(1) Construct a FTN from a given FSG.

(2) Construct an m * m transition matrix T from the FTN; where

m — is the total number of states in the FTN including the terminating state & ie. m is the number of nodes in the FTN.

t(i,j) — denotes the number of direct paths from state corresponding to nonterminal A_i to that associated with nonterminal A_j for a rule $A_i \longrightarrow a_{ij}A_j$ in the grammar, where a_{ij} is the terminal produced when traversing from A_i to A_j.

(3) Construct $T^k = T^{k-1} \cdot T$ (A.1)

for $k = 2,3,\ldots,n$

where n is an integer value when either of the following occurs :-

(a) all elements of T^n are zero. This implies that the grammar is nonrecursive. This follows because of the followings. By definition, element $t^k(i,j)$ of matrix T^k denotes the number of distinct strings, each of length k, that can be generated from the FTN by traversing from node A_i to node A_j. If there is no loop in the FTN (ie. the grammar is nonrecursive), then there must exist a value of n which exceeds the length of the longest string generated by the grammar. Hence, all elements of T^n are zero. The smallest value of n is, of course, equal to the longest string length plus one.

(b) two or more diagonal elements of T^n are nonzero which indicates that the grammar is recursive. This follows because all diagonal elements of T are zero due to the assumption of no self-loops in the FTN. The nonzero diagonal elements of T^n for some value of n thus indicate that there are transitions starting and ending at the same node associated with a diagonal element of T^n. This is the condition of looping in the FTN and consequently the grammar is recursive. In this case, the states of the FTN associated with diagonal elements of T^n whose values are nonzero define one or more of the loops in the FTN.

To illustrate the foregoing method, consider the following example.

Example A.1 Let $G_{A.1} = (V_N, \Sigma, R, £)$ be a FSG whose FTN is depicted in Fig. A.1 and where :-

$V_N = (A_1(=£), A_2, A_3, A_4, A_5, A_6, A_7)$

$\Sigma = (A, I, U)$

$R = A_1 \rightarrow UA_2 \quad A_3 \rightarrow IA_4 \quad A_5 \rightarrow AA_3 \quad A_6 \rightarrow AA_3$

$A_1 \rightarrow IA_6$ $A_3 \rightarrow I$ $A_5 \rightarrow AA_7$ $A_6 \rightarrow UA_5$

$A_2 \rightarrow AA_3$ $A_4 \rightarrow UA_5$ $A_5 \rightarrow A$ $A_7 \rightarrow I$

$A_2 \rightarrow IA_4$ $A_4 \rightarrow U$ $A_5 \rightarrow I$

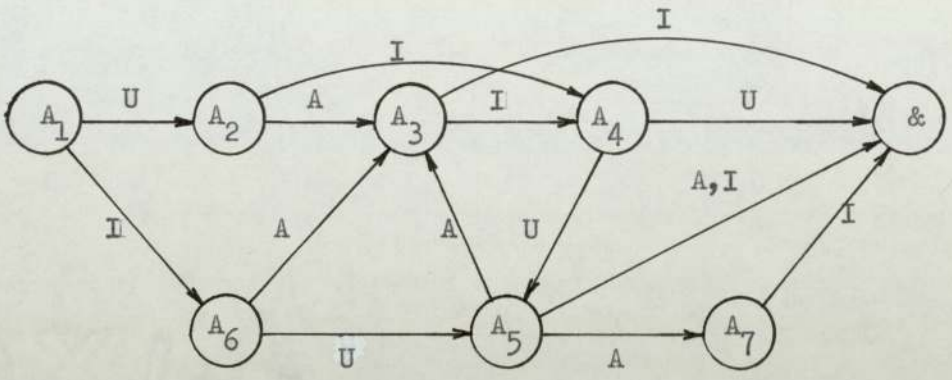


Fig. A.1 A FTN of the grammar $G_{A.1}$

First, a 8*8 T matrix is constructed as shown below.

$$T = \begin{array}{c} \\ \text{OUTGOING STATES} \end{array} \begin{array}{c} \begin{array}{cccccccc} \text{INCOMING STATES} & & & & & & & \\ A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 & \& \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ \& \end{array}$$

The next step is to find a value of n which satisfies either condition (a) or (b). As shown in the matrix T^3 below, diagonal elements

of T^3 are no longer zero. Since elements (3,3), (4,4) and (5,5) are nonzero, nodes A_3, A_4, and A_5 are in the loop. This is confirmed by the inspection of the graph. From the above results where condition (b) is satisfied, it can be concluded that $G_{A.1}$ is recursive.

T^3 =

OUTGOING STATES	INCOMING STATES: A_1	A_2	A_3	A_4	A_5	A_6	A_7	&
A_1	0	0	1	2	1	0	1	5
A_2	0	0	1	0	1	0	1	3
A_3	0	0	1	0	0	0	1	2
A_4	0	0	0	1	0	0	0	2
A_5	0	0	0	0	1	0	0	1
A_6	0	0	0	1	1	0	0	3
A_7	0	0	0	0	0	0	0	0
&	0	0	0	0	0	0	0	0

Finally, the computation of the number of distinct strings whose length do not exceed n follows immediately from the definitions of T and T^n. That is, the number of distinct strings of lengths $\leq n$ derivable from a given FSG is equal to :

$$\sum_{k=1}^{n} t^k(1 ,\&) \qquad \text{(A.2)}$$

As an example, for the grammar $G_{A.1}$, the number of distinct strings of lengths ≤ 3 is equal to $0 + 0 + 5 = 5$.

APPENDIX B

TABLE OF SIGNIFICANCE VALUES* OF SYMBOLS

Symbols	Values	Symbols	Values	Symbols	Values
A	1	U	21	j	-10
B	2	V	22	k	-11
C	3	W	23	ℓ	-12
D	4	X	24	m	-13
E	5	Y	25	n	-14
F	6	6	26	o	-15
G	7	7	27	p	-16
H	8	8	28	q	-17
I	9	9	29	r	-18
J	10	0	30	s	-19
K	11	@	100	t	-20
L	12	a	-1	u	-21
M	13	b	-2	v	-22
N	14	c	-3	w	-23
O	15	d	-4	x	-24
P	16	e	-5	y	-25
Q	17	f	-6	1	-26
R	18	g	-7	2	-27
S	19	h	-8	3	-28
T	20	i	-9	4	-29
				5	-30

*The above values are applicable to symbols generated by the FE of reference 96 .

APPENDIX C

TRAINING SET OF SYMBOL STRINGS

The followings are a training set of 100 symbol strings obtained from the FE of reference 96. The strings represent ten digits 'ZERO' to 'NINE' spoken by a single talker (A.J.P.) with ten repetitions in each digit. Values of significance of various symbols are provided in appendix B.

ONE	TWO	THREE
Pi	iD	fF
MkC	ℓCd	c
Lj	p	d
NmD	k	BhE
Nℓ E	m	Cc
Nk	ℓ	He
Pm	Ck	fF
Km	nEd	BhE
Nk	n	EiF
Lj	nD	f

FOUR	FIVE	SIX
JoC	BnRj	Bn@dCc
HrC	HjEℓPi	Fℓ@cCe
CnFc	CgCnRk	Cj@Bd
FnE	FoOg	Ck@c
GoF	EmOh	bCj@Ec
DpF	HpMk	Bi@Cc

FOUR(Cont.)

IoGe

HoB

FℓEd

LtH

FIVE(Cont.)

IrRh

BfEnQk

HqRi

EdCnSc

SIX(Cont.)

Ch@CfD

Fk@Cc

Gk@f

Bj@Db

SEVEN

DℓDd

Gn

FeEℓDf

Co

bFmCeC

BcEℓ

HeFp

BcEo

Eq

CcDpC

EIGHT

Gj@Bf

Ff@e

De@g

GjHj

CiFk

EkGg

Cd@Ej

Jh@ℓ

DiCcEd

Ih@b

NINE

Fℓ

Iℓ

Lk

IhC

Jg

Lk

Jh

Hg

Ji

IℓE

ZERO

CcDmF

CcDiEgE

BdFℓFeG

DhFiG

DeDhGe

ZERO(Cont.)

EfDhDeEd

FhEhE

EgDeC

EfEgC

FeDiF

APPENDIX D

RECOGNITION SET OF SYMBOL STRINGS

The following presents a recognition set of 500 symbol strings representing ten digits 'ZERO' to 'NINE'. Each digit is uttered 50 times by a single speaker (A.J.P.) using the FE of reference 96. Appendix B gives the values of significance of various symbols appearing in the recognition set.

ONE				TWO	
Nk	Mℓ	Lm	qA	Bℓ	q
Ph	Pg	Mℓ	o	ℓ	Bm
Mi	Nk	Mℓ	p	BoC	Bk
Lj	Ni	Mj	n	o	Eo
Pj	Nh	Nk	o	Bℓ	Ei
Nj	Mh	Ik	Bp	Do	g
Mi	Rk	Oi	o	q	CℓD
Lh	Nj	Pℓ	Bn	r	Bo
LkC	Pg	MiE	q	m	m
Mg	Kh	Oe	n	Bp	k
Mf	Kf		r	o	
Oh	Hd		o	i	
Oj	Nh		q	Bm	
Om	Kh		n	p	
Nℓ	Qh		Bp	t	
Pℓ	Og		o	s	
Nh	Kn		o	Eℓ	
Oj	Nm		q	m	
Ke	Ip		m	nD	
LkD	eKj		Dn	i	

THREE		FOUR		FIVE	
Bℓ	Ff	GwB	HoC	EdDkPc	HeCjN
ℓ	DiB	BxE	EpE	GdDmR	IoPi
FgCa	Ei	IuD	JoD	NhDkR	IgCjM
cDiA	GhC	EuD	jCiE	LgEkP	GqPc
Hk	Cg	DvE	CoF	QiCℓS	Eℓ0i
CdCh	bDd	EwE	JqI	JgDjQa	BℓPc
f	bDj	HrF	IqB	CfDmRe	BdDℓ0j
DfD	d	NwE	IpD	NjDi0	EjNe
CiC	BgB	LvG	JtF	GfCkQ	CoP
Cj	BgD	KuE	HqE	JiDkPf	EmQd
iDf	EhA	EsD	HrJ	GsSf	HnQe
e	FgCjE	PvC	HℓDc	GfCjN	FdDnQd
EgCf	Cg	BsD	nC	JhCℓT	GfCmNh
JhCg	BgB	sC	MmCfG	LiFmSdD	DnPg
h	FjC	FvE	EpF	EmEiQk	DcDoQi
CcDhD	CdDd	ItE	EoH	BjCjQh	DoQg
Hg	BgE	EtC	JoCe	IfDmTn	mPf
EgC	BhH	IrD	KoF	HnSj	HmMh
EeC	FℓD	EqD	CiB	CeEnRm	GeCmPf
DhC	GgG	HsF	FiCgE	EhDjPc	EeEnRk
bEf	EiF	IqE	HℓDiFc	mPg	GpSh
f	GiC	Fn	IjCiG	DgEnMi	oQf
EfEj	Eg	JsD	DnG	CeCkOj	GdCnPe
g	BhC	KuC	EpD	GdEoRf	eFℓNe
BiD	DiD	Gq	BsH	GmOc	EeDnQp

SIX		SEVEN		EIGHT	
DgCf@Dj	Bj@d	BcFi	Ch	Di@k	Ej@g
cFl@BgC	Be@bDc	EdGo	CeEjCe	e@h	EiCf
m@Dh	Bg@cD	Fp	Dj	Fi@h	De@f
Em@fH	f@c	DgGj	Cm	Ii@j	Ef@f
m@dA	Bg@cE	dIn	Gp	Gj@e	Cd@g
l@DfB	Fh@DcDd	BdFm	GnEd	Dg@@	Eh@e
Cm@BeE	Bk@Bd	DcDj	FmEe	FjC@m	Fh@Be
l@DfB	Df@EfC	EeFo	EmC	Cf@dCi	Fj@c
Dk@BeE	Dj@c	BcDk	HnDg	Df@l	Hh@f
i@Cc	Dj@fDc	BdIoB	CcCjCfC	Ce@Bi	Ch@Bf
Ei@Co	Cg@Cc	DfCkD	FcClCe	Bc@Bd	Bf@g
Bj@dDd	i@f	Dm	CcEp	Ee@Cg	ElGc
Dl@cDm	Dm@bB	CfFiCc	Gn	Ce@e	He@Bd
l@BcC	Fi@Cd	GfGp	Fn	Df@d	Gf@e
Bh@DcF	Bf@bC	El	Go	B@j	Fi@f
i@EcC	Ch@c	cDl	Fr	Bd@g	Cf@e
n@DhD	Fk@Cc	DeGo	Fn	Ca@e	Hi@BeCg
Cn@gD	Ci@Bc	CfHm	cDi	Bd@e	Je@f
Fj@eB	Eh@DgE	JoC	cHn	bB@d	Dk@Bj
k@dG	f@A	CdFo	IcClDc	b@d	Eg@bCh
Em@CgG	Gi@Ce	FdDj	ClC	Bg@a	Fg@f
Ci@bF	BdEj@h	bDk	BdGmA	De@cCg	Fg@Ch
f@fD	Eg@d	EdDi	cEk	Cf@d	Gg@g
k@G	Bg@d	FeEk	Fq	Fg@c	Hm@Be
bCd@Bf	Ch@Dc	FdFlDd	DcFp	Ig@eB	Ek@Bg

NINE		ZERO	
IoC	HfEi	GfGf	ChFgCcD
Hℓ	Mk	IgDgD	CfCjGfGc
Hh	OgDg	FhEg	eCgCcB
Kn	FeEiC	KmF	BiEhE
PtC	Hk	GhCfF	jD
Hk	JkC	EmB	ChEhE
bGj	Lk	BjF	DfDhD
Hh	Ii	GiChA	DnGeB
Hf	IeEh	GdCiC	jE
Eo	Kℓ	GjEfD	DkE
HℓD	KeEj	BeA	ChFf
Lk	HnC	HhCe	EcEhDd
Fm	KℓE	BdEhC	IdCfDe
Fk	OmE	HnH	HiB
Fℓ	ImE	ChCgC	EfDcDi
Eh	JnD	GℓF	BeCf
KfDg	Em	EjH	GhDfGd
Hg	Nk	ChF	gEcD
Ik	Jk	CfDc	GiGf
BmE	JℓC	GjG	IgEf
In	Mo	EeCfD	FgEfD
GgCc	GdCi	EfCc	EhFhF
KℓD	Nℓ	CiFgG	FdDe
Gℓ	IkB	eDdD	GdEcCiC
Ii	Hk	FcDgCa	GdEjCeE

APPENDIX E

RULES OF THE INFERRED GRAMMARS

The following presents rules of FSG's, weighted and non-weighted CFG's constructed directly from the training set in appendix C. The appropriate inference algorithms used can be found in chapters 3 and 4 . The set of terminal symbols is assumed to consist of all symbols appearing in appendix B. All other symbols including the start symbol £ are in the set of nonterminals.

E.1 Rules of the FSG's

ONE

£ → PA_2	£ → KA_9	A_3 → kA_4	A_6 → mA_7	A_7 → D
£ → MA_3	A_2 → i	A_4 → C	A_6 → ℓA_8	A_8 → E
£ → LA_5	A_2 → m	A_5 → j	A_6 → k	A_9 → m
£ → NA_6				

TWO

£ → iA_2	£ → m	£ → nA_6	A_3 → CA_4	A_6 → EA_7
£ → ℓA_3	£ → ℓ	£ → n	A_4 → d	A_6 → D
£ → p	£ → CA_5	A_2 → D	A_5 → k	A_7 → d
£ → k				

THREE

£ → fA_2	£ → BA_3	£ → EA_7	A_3 → hA_4	A_6 → ℓ
£ → c	£ → CA_5	£ → f	A_4 → E	A_7 → iA_8
£ → d	£ → HA_6	A_2 → F	A_5 → c	A_8 → F

FOUR

£ → JA_2 £ → LA_{19} A_5 → C A_{11} → oA_3 A_{18} → d

£ → HA_4 A_2 → oA_3 A_6 → nA_7 A_{12} → pA_{13} A_{19} → tA_{20}

£ → CA_6 A_3 → C A_7 → FA_8 A_{13} → F A_{20} → H

£ → FA_9 A_3 → F A_8 → c A_{14} → oA_{15}

£ → GA_{11} A_3 → B A_9 → nA_{10} A_{15} → GA_{16}

£ → DA_{12} A_4 → rA_5 A_9 → ℓA_{17} A_{16} → e

£ → IA_{14} A_4 → oA_3 A_{10} → E A_{17} → EA_{18}

FIVE

£ → BA_2 A_3 → QA_{23} A_6 → EA_7 A_{14} → OA_{15} A_{21} → RA_4

£ → HA_5 A_3 → SA_{26} A_7 → ℓA_8 A_{15} → g A_{22} → EA_7

£ → CA_{10} A_4 → j A_7 → nA_3 A_{15} → h A_{23} → k

£ → FA_{13} A_4 → k A_8 → PA_9 A_{16} → mA_{17} A_{24} → RA_4

£ → EA_{16} A_4 → h A_9 → i A_{16} → dA_{25} A_{25} → CA_{12}

£ → IA_{20} A_4 → i A_{10} → gA_{11} A_{17} → OA_{15} A_{26} → c

A_2 → nA_3 A_5 → jA_6 A_{11} → CA_{12} A_{18} → MA_{19}

A_2 → fA_{22} A_5 → pA_{18} A_{12} → nA_3 A_{19} → k

A_3 → RA_4 A_5 → qA_{24} A_{13} → oA_{14} A_{20} → rA_{21}

SIX

£ → BA_2 A_3 → $@A_4$ A_6 → c A_{10} → kA_{13} A_{15} → CA_{16}

£ → FA_7 A_4 → dA_5 A_6 → e A_{10} → hA_{19} A_{16} → jA_{11}

£ → CA_{10} A_4 → cA_9 A_6 → fA_{20} A_{11} → $@A_4$ A_{17} → c

£ → bA_{15} A_4 → BA_{12} A_7 → ℓA_8 A_{12} → d A_{18} → $@A_4$

£ → GA_{21} A_4 → EA_{17} A_7 → kA_{13} A_{13} → $@A_{14}$ A_{19} → $@A_4$

A_2 → nA_3 A_4 → CA_6 A_8 → $@A_4$ A_{14} → c A_{20} → D

A_2 → iA_{18} A_4 → DA_{22} A_9 → CA_6 A_{14} → CA_6 A_{21} → KA_{13}

A_2 → jA_{11} A_5 → CA_6 A_{10} → jA_{11} A_{14} → f A_{22} → b

SEVEN

£ → DA_2 £ → EA_{20} A_7 → EA_8 A_{12} → CA_{13} A_{17} → o

£ → GA_5 A_2 → ℓA_3 A_7 → FA_{19} A_{13} → eA_{14} A_{18} → eA_7

£ → FA_6 A_3 → DA_4 A_8 → ℓA_3 A_{14} → C A_{19} → p

£ → CA_9 A_4 → d A_9 → o A_{15} → cA_{16} A_{20} → q

£ → bA_{10} A_4 → f A_9 → c A_{16} → EA_{17} A_{21} → pA_{22}

£ → BA_{15} A_5 → n A_{10} → FA_{11} A_{16} → DA_{21} A_{22} → C

£ → HA_{18} A_6 → eA_7 A_{11} → mA_{12} A_{17} → ℓ

EIGHT

£ → GA_2 A_3 → $@A_4$ A_8 → g A_{12} → dA_{18} A_{19} → j

£ → FA_6 A_3 → HA_{11} A_8 → ℓ A_{13} → FA_{14} A_{19} → d

£ → DA_9 A_4 → BA_5 A_8 → b A_{13} → CA_{22} A_{20} → hA_{21}

£ → CA_{12} A_4 → EA_{19} A_9 → EA_{10} A_{14} → k A_{21} → $@A_8$

£ → EA_{15} A_5 → f A_9 → iA_{13} A_{15} → kA_{16} A_{22} → cA_{23}

£ → JA_{20} A_6 → fA_7 A_{10} → $@A_8$ A_{16} → GA_{17} A_{23} → EA_{19}

£ → IA_{24} A_7 → $@A_8$ A_{11} → j A_{17} → g A_{24} → hA_{21}

A_2 → jA_3 A_8 → e A_{12} → iA_{13} A_{18} → $@A_4$

NINE

£ → FA_2 £ → HA_7 A_3 → ℓA_8 A_6 → h

£ → IA_3 A_2 → ℓ A_4 → k A_6 → i

£ → LA_4 A_3 → ℓ A_5 → C A_7 → g

£ → JA_6 A_3 → hA_5 A_6 → g A_8 → E

ZERO

$\pounds \rightarrow CA_2$	$A_4 \rightarrow eA_{14}$	$A_{11} \rightarrow \ell A_{12}$	$A_{16} \rightarrow DA_{22}$	$A_{22} \rightarrow eA_{18}$
$\pounds \rightarrow BA_9$	$A_5 \rightarrow F$	$A_{11} \rightarrow iA_{17}$	$A_{16} \rightarrow EA_7$	$A_{23} \rightarrow d$
$\pounds \rightarrow DA_{15}$	$A_6 \rightarrow EA_7$	$A_{12} \rightarrow FA_{13}$	$A_{17} \rightarrow G$	$A_{24} \rightarrow hA_{16}$
$\pounds \rightarrow EA_{20}$	$A_6 \rightarrow F$	$A_{13} \rightarrow eA_{14}$	$A_{18} \rightarrow DA_4$	$A_{24} \rightarrow eA_{18}$
$\pounds \rightarrow FA_{24}$	$A_7 \rightarrow gA_8$	$A_{14} \rightarrow G$	$A_{18} \rightarrow EA_{23}$	$A_{25} \rightarrow E$
$A_2 \rightarrow cA_3$	$A_7 \rightarrow hA_{25}$	$A_{14} \rightarrow C$	$A_{19} \rightarrow e$	$A_{26} \rightarrow DA_4$
$A_3 \rightarrow DA_4$	$A_8 \rightarrow E$	$A_{15} \rightarrow hA_{16}$	$A_{20} \rightarrow fA_{21}$	
$A_4 \rightarrow mA_5$	$A_8 \rightarrow C$	$A_{15} \rightarrow eA_{18}$	$A_{20} \rightarrow gA_{26}$	
$A_4 \rightarrow iA_6$	$A_9 \rightarrow dA_{10}$	$A_{16} \rightarrow FA_{11}$	$A_{21} \rightarrow DA_4$	
$A_4 \rightarrow hA_{16}$	$A_{10} \rightarrow FA_{11}$	$A_{16} \rightarrow GA_{19}$	$A_{21} \rightarrow EA_7$	

E.2 Rules of the CFG's (weighted)

ONE

$\pounds \rightarrow A_1A_2$	$\pounds \rightarrow A_8A_4$	$A_3 \rightarrow M$	$A_8 \rightarrow N$	$A_{13} \rightarrow K$
$\pounds \rightarrow A_{36}A_5$	$\pounds \rightarrow A_1A_9$	$A_4 \rightarrow k$	$A_9 \rightarrow m$	$A_{36} \rightarrow A_3A_4$
$\pounds \rightarrow A_6A_7$	$\pounds \rightarrow A_{13}A_9$	$A_5 \rightarrow C$	$A_{10} \rightarrow D$	$A_{37} \rightarrow A_8A_9$
$\pounds \rightarrow A_{37}A_{10}$	$A_1 \rightarrow P$	$A_6 \rightarrow L$	$A_{11} \rightarrow \ell$	$A_{38} \rightarrow A_8A_{11}$
$\pounds \rightarrow A_{38}A_{12}$	$A_2 \rightarrow i$	$A_7 \rightarrow j$	$A_{12} \rightarrow E$	

TWO

$\pounds \rightarrow A_1A_2$	$\pounds \rightarrow A_3$	$A_1 \rightarrow i$	$A_6 \rightarrow p$	$A_{36} \rightarrow A_3A_4$
$\pounds \rightarrow A_{36}A_5$	$\pounds \rightarrow A_4A_7$	$A_2 \rightarrow D$	$A_7 \rightarrow k$	$A_{37} \rightarrow A_9A_{10}$
$\pounds \rightarrow A_6$	$\pounds \rightarrow A_{37}A_5$	$A_3 \rightarrow \ell$	$A_8 \rightarrow m$	
$\pounds \rightarrow A_7$	$\pounds \rightarrow A_9$	$A_4 \rightarrow C$	$A_9 \rightarrow n$	
$\pounds \rightarrow A_8$	$\pounds \rightarrow A_9A_2$	$A_5 \rightarrow d$	$A_{10} \rightarrow E$	

THREE

£ → A_1A_2 £ → A_9A_{10} A_3 → c A_8 → C A_{37} → $A_{11}A_2$

£ → A_3 £ → A_7A_{37} A_4 → d A_9 → H

£ → A_4 £ → A_1 A_5 → B A_{10} → e

£ → A_5A_{36} A_1 → f A_6 → h A_{11} → i

£ → A_8A_3 A_2 → F A_7 → E A_{36} → A_6A_7

FOUR

£ → $A_{36}A_3$ £ → $A_{48}A_4$ A_9 → E A_{18} → L A_{43} → $A_{13}A_2$

£ → $A_{37}A_3$ A_1 → J A_{10} → G A_{19} → t A_{44} → $A_{43}A_{10}$

£ → $A_{39}A_8$ A_2 → o A_{11} → D A_{36} → A_1A_2 A_{45} → A_4A_2

£ → A_7A_{40} A_3 → C A_{12} → p A_{37} → A_4A_5 A_{46} → A_7A_{16}

£ → $A_{41}A_7$ A_4 → H A_{13} → I A_{38} → A_3A_6 A_{47} → $A_{46}A_9$

£ → $A_{11}A_{42}$ A_5 → r A_{14} → e A_{39} → $A_{38}A_7$ A_{48} → $A_{18}A_{19}$

£ → $A_{44}A_{14}$ A_6 → n A_{15} → B A_{40} → A_6A_9

£ → $A_{45}A_{15}$ A_7 → F A_{16} → ℓ A_{41} → $A_{10}A_2$

£ → $A_{47}A_{17}$ A_8 → c A_{17} → d A_{42} → $A_{12}A_7$

FIVE

£ → $A_{37}A_4$ A_4 → j A_{17} → h A_{38} → A_7A_8 A_{51} → $A_{50}A_{12}$

£ → A_5A_{41} A_5 → H A_{18} → p A_{39} → $A_{38}A_9$ A_{52} → $A_{21}A_3$

£ → $A_{10}A_{45}$ A_6 → E A_{19} → M A_{40} → A_6A_{39} A_{53} → $A_{52}A_{17}$

£ → $A_{13}A_{47}$ A_7 → ℓ A_{20} → I A_{41} → A_4A_{40} A_{54} → A_2A_{23}

£ → A_6A_{49} A_8 → P A_{21} → r A_{42} → A_2A_3 A_{55} → $A_{54}A_{12}$

£ → A_5A_{51} A_9 → i A_{22} → f A_{43} → $A_{42}A_{12}$ A_{56} → A_6A_{55}

£ → $A_{20}A_{53}$ A_{10} → C A_{23} → Q A_{44} → $A_{10}A_{43}$ A_{57} → $A_{22}A_{56}$

£ → A_1A_{57} A_{11} → g A_{24} → q A_{45} → $A_{11}A_{44}$ A_{58} → $A_{24}A_3$

£ → A_5A_{59} A_{12} → k A_{25} → d A_{46} → $A_{14}A_{15}$ A_{59} → $A_{58}A_9$

£ → A_6A_{63} A_{13} → F A_{26} → S A_{47} → $A_{46}A_{11}$ A_{60} → A_2A_{26}

A_1 → B A_{14} → o A_{27} → c A_{48} → $A_{16}A_{15}$ A_{61} → $A_{60}A_{27}$

A_2 → n A_{15} → O A_{36} → A_1A_2 A_{49} → $A_{48}A_{17}$ A_{62} → $A_{10}A_{61}$

A_3 → R A_{16} → m A_{37} → $A_{36}A_3$ A_{50} → $A_{18}A_{19}$ A_{63} → $A_{25}A_{62}$

SIX

£ → $A_{39}A_6$
£ → A_7A_{43}
£ → A_5A_{46}
£ → $A_{48}A_6$
£ → $A_{12}A_{51}$
£ → A_1A_{54}
£ → $A_{58}A_{17}$
£ → A_7A_{60}
£ → $A_{18}A_{59}$
£ → $A_{50}A_{12}$
A_1 → B
A_2 → n
A_3 → @
A_4 → d
A_5 → C
A_6 → c
A_7 → F
A_8 → ℓ
A_9 → e
A_{10} → j
A_{11} → k
A_{12} → b
A_{13} → E
A_{14} → i
A_{15} → h
A_{16} → f
A_{17} → D
A_{18} → G
A_{36} → A_1A_2
A_{37} → $A_{36}A_3$
A_{38} → $A_{37}A_4$
A_{39} → $A_{38}A_5$
A_{40} → A_8A_3
A_{41} → $A_{40}A_6$
A_{42} → $A_{41}A_5$
A_{43} → $A_{42}A_9$
A_{44} → $A_{10}A_3$
A_{45} → $A_{44}A_1$
A_{46} → $A_{45}A_4$
A_{47} → $A_{11}A_3$
A_{48} → A_5A_{47}
A_{49} → A_5A_{44}
A_{49} → A_1A_{44}
A_{50} → $A_{49}A_{13}$
A_{50} → $A_{49}A_{17}$
A_{51} → $A_{50}A_6$
A_{52} → $A_{14}A_3$
A_{53} → $A_{52}A_5$
A_{54} → $A_{53}A_6$
A_{55} → $A_{15}A_3$
A_{56} → A_5A_{55}
A_{57} → $A_{56}A_5$
A_{58} → $A_{57}A_{16}$
A_{59} → $A_{47}A_5$
A_{59} → $A_{47}A_{16}$
A_{60} → $A_{59}A_6$

ZERO

£ → $A_{38}A_5$
£ → $A_{43}A_7$
£ → $A_{48}A_{13}$
£ → $A_{51}A_{13}$
£ → $A_{55}A_{12}$
£ → $A_{61}A_{10}$
£ → A_5A_{64}
£ → $A_{67}A_1$
£ → $A_{70}A_1$
£ → $A_{73}A_5$
A_1 → C
A_2 → c
A_3 → D
A_4 → m
A_5 → F
A_6 → i
A_7 → E
A_8 → g
A_9 → B
A_{10} → d
A_{11} → ℓ
A_{12} → e
A_{13} → G
A_{14} → h
A_{15} → f
A_{36} → A_1A_2
A_{37} → $A_{36}A_3$
A_{38} → $A_{37}A_4$
A_{39} → A_6A_7
A_{40} → A_3A_{39}
A_{41} → A_2A_{40}
A_{42} → A_1A_{41}
A_{43} → $A_{42}A_8$
A_{44} → $A_{11}A_5$
A_{45} → A_5A_{44}
A_{46} → $A_{10}A_{45}$
A_{47} → A_9A_{46}
A_{48} → $A_{47}A_{12}$
A_{49} → $A_{14}A_5$
A_{50} → A_3A_{49}
A_{51} → $A_{50}A_6$
A_{52} → $A_{14}A_{13}$
A_{53} → A_3A_{52}
A_{54} → $A_{12}A_{53}$
A_{55} → A_3A_{54}
A_{56} → A_3A_{14}
A_{57} → $A_{56}A_3$
A_{58} → $A_{15}A_{57}$
A_{59} → $A_{58}A_{12}$
A_{60} → A_7A_{59}
A_{61} → $A_{60}A_7$
A_{62} → $A_{14}A_7$
A_{63} → $A_{62}A_{14}$
A_{64} → $A_{63}A_7$
A_{65} → A_8A_3
A_{66} → A_7A_{65}
A_{67} → $A_{66}A_{12}$
A_{68} → A_7A_{15}
A_{69} → $A_{68}A_7$
A_{70} → $A_{69}A_8$
A_{71} → A_5A_{12}
A_{72} → $A_{71}A_3$
A_{73} → $A_{72}A_6$

SEVEN

£ → $A_{37}A_3$ A_1 → D A_{11} → o A_{38} → A_8A_2 A_{48} → A_7A_{47}

£ → A_4A_5 A_2 → ℓ A_{12} → b A_{39} → $A_{38}A_1$ A_{49} → A_8A_{11}

£ → A_6A_{41} A_3 → d A_{13} → m A_{40} → $A_{39}A_9$ A_{50} → $A_{15}A_{49}$

£ → $A_{10}A_{11}$ A_4 → G A_{14} → B A_{41} → A_7A_{40} A_{51} → A_1A_{17}

£ → $A_{45}A_{10}$ A_5 → n A_{15} → c A_{42} → A_6A_{13} A_{52} → $A_{15}A_{51}$

£ → $A_{14}A_{46}$ A_6 → F A_{16} → H A_{43} → $A_{42}A_{10}$ A_{53} → $A_{10}A_{52}$

£ → $A_{16}A_{48}$ A_7 → e A_{17} → p A_{44} → $A_{43}A_7$

£ → $A_{14}A_{50}$ A_8 → E A_{18} → q A_{45} → $A_{12}A_{44}$

£ → A_8A_{18} A_9 → f A_{36} → A_1A_2 A_{46} → $A_{15}A_{38}$

£ → $A_{53}A_{10}$ A_{10} → C A_{37} → $A_{36}A_1$ A_{47} → A_6A_{17}

EIGHT

£ → $A_{38}A_5$ A_2 → j A_{13} → k A_{37} → $A_{36}A_{10}$ A_{48} → $A_{47}A_{14}$

£ → $A_{40}A_7$ A_3 → @ A_{14} → E A_{38} → $A_{37}A_4$ A_{49} → $A_{48}A_2$

£ → $A_{42}A_9$ A_4 → B A_{15} → d A_{39} → A_5A_3 A_{50} → $A_{17}A_3$

£ → $A_{37}A_2$ A_5 → f A_{16} → J A_{40} → A_6A_{39} A_{51} → $A_{16}A_{50}$

£ → $A_{11}A_{44}$ A_6 → F A_{17} → h A_{41} → A_7A_3 A_{51} → $A_{20}A_{50}$

£ → $A_{14}A_{46}$ A_7 → e A_{18} → ℓ A_{42} → A_8A_{41} A_{52} → A_8A_{12}

£ → $A_{11}A_{49}$ A_8 → D A_{19} → c A_{43} → $A_{12}A_6$ A_{53} → $A_{52}A_{11}$

£ → $A_{51}A_{18}$ A_9 → g A_{20} → I A_{44} → $A_{43}A_{13}$ A_{54} → $A_{53}A_{19}$

£ → $A_{55}A_{15}$ A_{10} → H A_{21} → b A_{45} → $A_{13}A_1$ A_{55} → $A_{54}A_{14}$

£ → $A_{51}A_{21}$ A_{11} → C A_{36} → A_1A_2 A_{46} → $A_{45}A_9$

A_1 → G A_{12} → i A_{37} → $A_{36}A_3$ A_{47} → $A_{15}A_3$

NINE

£ → A_1A_2 £ → A_8A_6 A_2 → ℓ A_7 → C A_{12} → E

£ → A_3A_2 £ → $A_{10}A_9$ A_3 → I A_8 → J A_{36} → A_3A_6

£ → A_4A_5 £ → A_8A_{11} A_4 → L A_9 → g A_{37} → A_3A_2

£ → $A_{36}A_7$ £ → $A_{37}A_{12}$ A_5 → k A_{10} → H

£ → A_8A_9 A_1 → F A_6 → h A_{11} → i

E.3 Rules of the CFG's (nonweighted)

ONE

$£ \rightarrow A_1 A_2$	$£ \rightarrow A_8 A_4$	$A_3 \rightarrow M$	$A_8 \rightarrow N$	$A_{13} \rightarrow K$
$£ \rightarrow A_{36} A_5$	$£ \rightarrow A_1 A_9$	$A_4 \rightarrow k$	$A_9 \rightarrow m$	$A_{36} \rightarrow A_3 A_4$
$£ \rightarrow A_6 A_7$	$£ \rightarrow A_{13} A_9$	$A_5 \rightarrow C$	$A_{10} \rightarrow D$	$A_{37} \rightarrow A_8 A_9$
$£ \rightarrow A_{37} A_{10}$	$A_1 \rightarrow P$	$A_6 \rightarrow L$	$A_{11} \rightarrow \ell$	$A_{38} \rightarrow A_8 A_{11}$
$£ \rightarrow A_{38} A_{12}$	$A_2 \rightarrow i$	$A_7 \rightarrow j$	$A_{12} \rightarrow E$	

TWO

$£ \rightarrow A_1 A_2$	$£ \rightarrow A_3$	$A_1 \rightarrow i$	$A_6 \rightarrow p$	$A_{36} \rightarrow A_3 A_4$
$£ \rightarrow A_{36} A_5$	$£ \rightarrow A_4 A_7$	$A_2 \rightarrow D$	$A_7 \rightarrow k$	$A_{37} \rightarrow A_9 A_{10}$
$£ \rightarrow A_6$	$£ \rightarrow A_{37} A_5$	$A_3 \rightarrow \ell$	$A_8 \rightarrow m$	
$£ \rightarrow A_7$	$£ \rightarrow A_9$	$A_4 \rightarrow C$	$A_9 \rightarrow n$	
$£ \rightarrow A_8$	$£ \rightarrow A_9 A_2$	$A_5 \rightarrow d$	$A_{10} \rightarrow E$	

THREE

$£ \rightarrow A_1 A_2$	$£ \rightarrow A_9 A_{10}$	$A_3 \rightarrow c$	$A_8 \rightarrow C$	$A_{37} \rightarrow A_{11} A_2$
$£ \rightarrow A_3$	$£ \rightarrow A_7 A_{37}$	$A_4 \rightarrow d$	$A_9 \rightarrow H$	
$£ \rightarrow A_4$	$£ \rightarrow A_1$	$A_5 \rightarrow B$	$A_{10} \rightarrow e$	
$£ \rightarrow A_{36} A_7$	$A_1 \rightarrow f$	$A_6 \rightarrow h$	$A_{11} \rightarrow i$	
$£ \rightarrow A_8 A_3$	$A_2 \rightarrow F$	$A_7 \rightarrow E$	$A_{36} \rightarrow A_5 A_6$	

NINE

$£ \rightarrow A_1 A_2$	$£ \rightarrow A_8 A_6$	$A_2 \rightarrow \ell$	$A_7 \rightarrow C$	$A_{12} \rightarrow E$
$£ \rightarrow A_3 A_2$	$£ \rightarrow A_{10} A_9$	$A_3 \rightarrow I$	$A_8 \rightarrow J$	$A_{36} \rightarrow A_3 A_6$
$£ \rightarrow A_4 A_5$	$£ \rightarrow A_8 A_{11}$	$A_4 \rightarrow L$	$A_9 \rightarrow g$	$A_{37} \rightarrow A_3 A_2$
$£ \rightarrow A_{36} A_7$	$£ \rightarrow A_{37} A_{12}$	$A_5 \rightarrow k$	$A_{10} \rightarrow H$	
$£ \rightarrow A_8 A_9$	$A_1 \rightarrow F$	$A_6 \rightarrow h$	$A_{11} \rightarrow i$	

FOUR

£ → $A_{36}A_3$
£ → A_4A_{37}
£ → $A_{39}A_8$
£ → $A_{40}A_9$
£ → $A_{41}A_7$
£ → $A_{11}A_{42}$
£ → $A_{44}A_{14}$
£ → $A_{45}A_{15}$
£ → $A_{47}A_{17}$

£ → $A_{48}A_4$
A_1 → J
A_2 → o
A_3 → C
A_4 → H
A_5 → r
A_6 → n
A_7 → F
A_8 → c

A_9 → E
A_{10} → G
A_{11} → D
A_{12} → p
A_{13} → I
A_{14} → e
A_{15} → B
A_{16} → ℓ
A_{17} → d

A_{18} → L
A_{19} → t
A_{36} → A_1A_2
A_{37} → A_5A_3
A_{38} → A_3A_6
A_{39} → $A_{38}A_7$
A_{40} → A_7A_6
A_{41} → $A_{10}A_2$
A_{42} → $A_{12}A_7$

A_{43} → $A_{13}A_2$
A_{44} → $A_{43}A_{10}$
A_{45} → A_4A_2
A_{46} → A_7A_{16}
A_{47} → $A_{46}A_9$
A_{48} → $A_{18}A_{19}$

FIVE

£ → $A_{37}A_4$
£ → $A_{41}A_9$
£ → $A_{10}A_{45}$
£ → $A_{47}A_{11}$
£ → $A_{49}A_{17}$
£ → $A_{51}A_{12}$
£ → $A_{53}A_{17}$
£ → $A_{57}A_{12}$
£ → $A_{59}A_9$
£ → $A_{63}A_{27}$
A_1 → B
A_2 → n
A_3 → R

A_4 → j
A_5 → H
A_6 → E
A_7 → ℓ
A_8 → P
A_9 → i
A_{10} → C
A_{11} → g
A_{12} → k
A_{13} → F
A_{14} → o
A_{15} → O
A_{16} → m

A_{17} → h
A_{18} → p
A_{19} → M
A_{20} → I
A_{21} → r
A_{22} → f
A_{23} → Q
A_{24} → q
A_{25} → d
A_{26} → S
A_{27} → c
A_{36} → A_1A_2
A_{37} → $A_{36}A_3$

A_{38} → A_5A_4
A_{39} → $A_{38}A_6$
A_{40} → $A_{39}A_7$
A_{41} → $A_{40}A_8$
A_{42} → A_2A_3
A_{43} → $A_{10}A_{42}$
A_{44} → $A_{43}A_{12}$
A_{45} → $A_{11}A_{44}$
A_{46} → $A_{13}A_{14}$
A_{47} → $A_{46}A_{15}$
A_{48} → $A_{16}A_{15}$
A_{49} → A_6A_{48}
A_{50} → A_5A_{18}

A_{51} → $A_{50}A_{19}$
A_{52} → $A_{21}A_3$
A_{53} → $A_{20}A_{52}$
A_{54} → A_1A_{22}
A_{55} → $A_{54}A_6$
A_{56} → $A_{55}A_2$
A_{57} → $A_{56}A_{23}$
A_{58} → A_5A_{24}
A_{59} → $A_{58}A_3$
A_{60} → A_6A_{25}
A_{61} → $A_{60}A_{10}$
A_{62} → $A_{61}A_2$
A_{63} → $A_{62}A_{26}$

SIX

£ → $A_{39}A_6$
£ → $A_{43}A_9$
£ → $A_{46}A_4$
£ → $A_{48}A_6$
£ → $A_{12}A_{49}$
£ → $A_{52}A_6$
£ → $A_{56}A_{17}$
£ → $A_{59}A_6$
£ → $A_{58}A_{16}$
£ → $A_{62}A_{12}$
A_1 → B
A_2 → n
A_3 → @
A_4 → d
A_5 → C
A_6 → c
A_7 → F
A_8 → ℓ
A_9 → e
A_{10} → j
A_{11} → k
A_{12} → b
A_{13} → E
A_{14} → i
A_{15} → h
A_{16} → f
A_{17} → D
A_{18} → G
A_{36} → A_1A_2
A_{37} → $A_{36}A_3$
A_{38} → $A_{37}A_4$
A_{39} → $A_{38}A_5$
A_{40} → A_3A_6
A_{41} → A_8A_{40}
A_{42} → A_7A_{41}
A_{43} → $A_{42}A_5$
A_{44} → A_5A_{10}
A_{45} → $A_{44}A_3$
A_{46} → $A_{45}A_1$
A_{46} → $A_{45}A_{13}$
A_{47} → A_5A_{11}
A_{48} → $A_{47}A_3$
A_{49} → $A_{46}A_6$
A_{50} → $A_{14}A_3$
A_{51} → A_1A_{50}
A_{52} → $A_{51}A_5$
A_{53} → A_5A_{15}
A_{54} → $A_{53}A_3$
A_{55} → $A_{54}A_5$
A_{56} → $A_{55}A_{16}$
A_{57} → $A_{11}A_3$
A_{58} → A_7A_{57}
A_{58} → $A_{18}A_{57}$
A_{59} → $A_{58}A_5$
A_{60} → $A_{10}A_3$
A_{61} → A_1A_{60}
A_{62} → $A_{61}A_{17}$

SEVEN

£ → $A_{37}A_3$
£ → A_4A_5
£ → A_6A_{41}
£ → $A_{10}A_{11}$
£ → $A_{45}A_{10}$
£ → $A_{47}A_2$
£ → $A_{49}A_{17}$
£ → $A_{47}A_{11}$
£ → A_8A_{18}
£ → $A_{52}A_{10}$
A_1 → D
A_2 → ℓ
A_3 → d
A_4 → G
A_5 → n
A_6 → F
A_7 → e
A_8 → E
A_9 → f
A_{10} → C
A_{11} → o
A_{12} → b
A_{13} → m
A_{14} → B
A_{15} → c
A_{16} → H
A_{17} → p
A_{18} → q
A_{36} → A_1A_2
A_{37} → $A_{36}A_1$
A_{38} → A_8A_2
A_{39} → $A_{38}A_1$
A_{40} → $A_{39}A_9$
A_{41} → A_7A_{40}
A_{42} → $A_{10}A_7$
A_{43} → $A_{13}A_{42}$
A_{44} → A_6A_{43}
A_{45} → $A_{12}A_{44}$
A_{46} → $A_{14}A_{15}$
A_{47} → $A_{46}A_8$
A_{48} → $A_{16}A_7$
A_{49} → $A_{48}A_6$
A_{50} → $A_{10}A_{15}$
A_{51} → $A_{50}A_1$
A_{52} → $A_{51}A_{17}$

EIGHT

£ → $A_{38}A_5$
£ → $A_{40}A_7$
£ → $A_{42}A_9$
£ → $A_{37}A_2$
£ → $A_{44}A_{13}$
£ → $A_{14}A_{46}$
£ → $A_{49}A_2$
£ → $A_{51}A_{18}$
£ → $A_{55}A_{15}$
£ → $A_{51}A_{21}$
A_1 → G
A_2 → j
A_3 → @
A_4 → B
A_5 → f
A_6 → F
A_7 → e
A_8 → D
A_9 → g
A_{10} → H
A_{11} → C
A_{12} → i
A_{13} → k
A_{14} → E
A_{15} → d
A_{16} → J
A_{17} → h
A_{18} → ℓ
A_{19} → c
A_{20} → I
A_{21} → b
A_{36} → A_1A_2
A_{37} → $A_{36}A_3$
A_{37} → $A_{36}A_{10}$
A_{38} → $A_{37}A_4$
A_{39} → A_6A_5
A_{40} → $A_{39}A_3$
A_{41} → A_8A_7
A_{42} → $A_{41}A_3$
A_{43} → $A_{11}A_{12}$
A_{44} → $A_{43}A_6$
A_{45} → A_1A_9
A_{46} → $A_{13}A_{45}$
A_{47} → $A_{11}A_{15}$
A_{48} → $A_{47}A_3$
A_{49} → $A_{48}A_{14}$
A_{50} → $A_{17}A_3$
A_{51} → $A_{16}A_{50}$
A_{51} → $A_{20}A_{50}$
A_{52} → A_8A_{12}
A_{53} → $A_{52}A_{11}$
A_{54} → $A_{53}A_{19}$
A_{55} → $A_{54}A_{14}$

ZERO

£ → $A_{38}A_5$
£ → $A_{40}A_7$
£ → $A_{45}A_{13}$
£ → $A_{48}A_{13}$
£ → $A_{50}A_{12}$
£ → $A_{53}A_{10}$
£ → $A_{56}A_7$
£ → $A_{59}A_1$
£ → A_7A_{62}
£ → A_5A_{65}
A_1 → C
A_2 → c
A_3 → D
A_4 → m
A_5 → F
A_6 → i
A_7 → E
A_8 → g
A_9 → B
A_{10} → d
A_{11} → ℓ
A_{12} → e
A_{13} → G
A_{14} → h
A_{15} → f
A_{36} → A_1A_2
A_{37} → $A_{36}A_3$
A_{38} → $A_{37}A_4$
A_{38} → $A_{37}A_6$
A_{39} → $A_{38}A_7$
A_{40} → $A_{39}A_8$
A_{41} → $A_{11}A_5$
A_{42} → A_5A_{41}
A_{43} → $A_{10}A_{42}$
A_{44} → A_9A_{43}
A_{45} → $A_{44}A_{12}$
A_{46} → A_3A_{14}
A_{46} → A_3A_{14}
A_{47} → $A_{46}A_5$
A_{47} → $A_{46}A_{13}$
A_{47} → $A_{46}A_3$
A_{48} → $A_{47}A_6$
A_{48} → $A_{47}A_{12}$
A_{49} → $A_{12}A_{47}$
A_{50} → A_3A_{49}
A_{51} → $A_{15}A_{48}$
A_{52} → A_7A_{51}
A_{53} → $A_{52}A_7$
A_{54} → A_5A_{14}
A_{55} → $A_{54}A_7$
A_{56} → $A_{55}A_{14}$
A_{57} → A_3A_{12}
A_{58} → A_8A_{57}
A_{59} → A_7A_{58}
A_{60} → A_7A_8
A_{61} → $A_{60}A_1$
A_{62} → $A_{15}A_{61}$
A_{63} → A_3A_6
A_{64} → $A_{63}A_5$
A_{65} → $A_{12}A_{64}$

REFERENCES

1. M.B. HERSCHER and R.B. COX, 'Talk to the computer : an adaptive isolated-word speech recognition system', NAVMIRO Manufacturing Technology Bulletin, no. 45, August 1973.

2. D.R. HILL, 'Man-machine interaction using speech', in F.L.Alt,M. Rubinoff and M.C.Youits, Eds., Advances in Computers, vol. II, pp. 165-230, New York : Academic Press, 1971.

3. W.A. LEA, 'Establishing the value of voice communication with computers', IEEE Transactions on Audio and Electroacoustics, vol. AU-16, pp. 184-197, June 1968.

4. N. LINDGREN, 'Speech - man's natural communication', IEEE Spectrum, vol. 4, pp. 75-86, June 1967.

5. D.R. REDDY, 'Speech recognition by machine : a review', Proceedings of the IEEE, vol. 64, no. 4, pp. 501-531, April 1976.

6. 'A voice-controlled computer terminal for the severely disabled', Information note of Scope Electronics Inc., Reston, Virginia 22090.

7. J.W. GLENN,'Machines you can talk to', Machine Design, pp. 72-75, May 1, 1975.

8. 'Voice recognition systems', A brochure of EMI Threshold Ltd., Hayes, Middlesex, England.

9. 'Tell your cartons where to get off ', Traffic Management, pp. 24-27, Denver, CO : Cahners Publisher, February 1975.

10. T.B. MARTIN, 'Practical applications of voice input to machines', Proceedings of the IEEE, vol. 64, no. 4, pp. 487-501, April 1976.

11. M.J. UNDERWOOD, 'Machine that understand speech', The radio and electronic engineer, vol. 47, no. 819, pp. 368-376, August/September 1977.

12. R.O. DUDA and P.E. HART, Pattern Classification and Scene Analysis, New York : John Wiley & Sons, Inc., 1973.

13. S. CHIBA, 'Spoken word recognition by multiple linear separation', The 6th International Congress on Acoustics, Tokyo, Japan, pp. B123-B126, August 21-28, 1968.

14. J.H. KING, Jr. and C.J. TUNIS, 'Some experiments in spoken word recognition', IBM Journal, pp. 65-79, January 1966.

15. T.G. von KELLER,'An on-line recognition system for spoken digits', Journal of the Acoustical Society of America, vol. 49, no.4 (part 2), pp. 1288-1296, 1971.

16. S. INOUE and A. KUREMATSU, 'Speech recognition with time-normalized frequency pattern', The 6th International Congress on Acoustics, Tokyo, Japan, pp. B127-B130, August 21-28, 1968.

17. L.C.W. POLS, 'Real-time recognition of spoken words', IEEE Transactions on Computers, vol. C-20, pp. 972-978, September 1971.

18. C.F. TEACHER, H.G. KELLETT and L. FOCHT, 'Experimental limited vocabulary, speech recognizer', IEEE Transactions on Audio and Electroacoustics, vol. AU-15, no. 3, September 1967.

19. G.M. WHITE, 'Simple techniques for transforming speech to quasi-phoneme strings', Speech Communication Seminar, Stockholm, August 1-3, 1974.

20. W. BEZDEL and J.S. BRIDLE, 'Speech recognition using zero-crossing measurements and sequence information', Proceedings IEE, vol. 116, no. 4, pp. 617-623, April 1969.

21. M.R. SAMBUR and L.R. RABINER, 'A speaker-independent digit-recognition system', BSTJ vol. 54, pp. 81-102, January 1975.

22. J.L. FLANAGAN, Speech analysis, synthesis and perception, New York : Academic Press, 1972.

23. K.H. DAVID, R. BIDDULPH and S. BALASHEK, 'Automatic recognition of spoken digits', Journal of the Acoustical Society of America, vol. 24, pp. 637-642, 1952.

24. G.D. EWING, 'Computer recognition of speech using zero-crossing information', IEEE Transactions on Audio and Electroacoustics, vol. AU-17, no. 1, pp. 37-40, March 1969.

25. R. de MORI, 'Speech analysis and recognition by computer using zero-crossing information', Acoustica, vol. 25,pp. 269-279, 1971.

26. B.S. ATAL and S.L. HANAUER, 'Speech analysis and synthesis by linear prediction of the speech wave', Journal of the Acoustical Society of America, vol. 50, no. 2 (part 2), pp. 637-655, 1971.

27. R.G. CRICHTON and F. FALLSIDE, 'Linear prediction model of speech production with applications to deaf speech training', Proceedings IEE, vol. 121, no. 8, pp. 865-873, August 1974.

28. P. DENES and M.V. MATHEWS, 'Spoken digit recognition using time-frequency pattern matching', Journal of the Acoustical Society of America, vol. 32, pp. 1450-1455, November 1960.

29. J.P. HATON, 'A practical application of a real-time isolated-word recognition system using syntactic constraints', IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-22, pp.416-419, December 1974.

30. D.E. WALKER, 'Speech understanding through syntactic and semantic analysis', IEEE Transactions on Computers, vol. C-25, no. 4, pp. 432-439, April 1976.

31. H. DUDLEY and S. BALASHEK, 'Automatic recognition of phonetic patterns in speech', Journal of the Acoustical Society of America, vol. 30, pp. 721-739, August 1958.

32. W.J. STEINGRANDT and S.S. YAU, 'Sequential feature extraction for waveform recognition', Spring Joint Computer Conference, pp. 65-76, 1970.

33. L.W. FUNG and K.S. FU, 'Maximum-likelihood syntactic decoding', IEEE Transactions on Information Theory, vol. IT-21, no. 4, pp. 423-430, July 1975.

34. G.R. DOWLING and P.A.V. HALL, 'Elastic template matching in speech recognition, using linguistic information', Second International Joint Conference on Pattern Recognition, Copenhagen, pp. 249-250, August 13-15, 1974.

35. G.R. DOWLING, 'A trainable recognizer for spoken sentences', Proceedings of International Computer Symposium, vol. I, pp. 386-391, 1975.

36. T.B. MARTIN, 'Automatic recognition of a limited vocabulary in continuous speech', Ph.D. dissertion, Department of Electrical Engineering, University of Pensylvania, 1970.

37. K.S. FU, Sequential Methods in Pattern Recognition and Machine Learning, New York : Academic Press, 1968.

38. K. FUKUNAGA, Introduction to Statistical Pattern Recognition, New York : Academic Press, 1972.

39. H.C. ANDREWS, Introduction to Mathematical Techniques in Pattern Recognition, New York : Wiley, 1972.

40. R. de MORI, 'A descriptive technique for automatic speech recognition', IEEE Transactions on Audio and Electroacoustics, vol. AU-21, no. 2, pp. 89-100, April 1973.

41. N. CHOMSKY, Aspects of the Theory of Syntax, Cambridge, Mass. : MIT Press, 1964.

42. K.S. FU and T.L. BOOTH, 'Grammatical inference : introduction and survey', IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-5, no. 1, pp. 95-111, January 1975 (Part I) and vol. SMC-5, no. 4, pp. 409-423, July 1975 (Part II).

43. K.S. FU and P.H. SWAIN, 'On syntactic pattern recognition', in J.T.Tou, Ed., Software Engineering, vol. 2, New York : Academic Press, 1971.

44. K.S. FU, Syntactic Methods in Pattern Recognition, New York : Academic Press, 1974.

45. R.S. LEDLEY, L.S. ROTOLO, T.J. GOLAB, J.D. JACOBSEN, M.D. GINSBURG and J.B. WILSON, 'FIDAC : Film input to digital automatic computer and associated syntax-directed pattern recognition programming system', in J.T. Tippett et al., eds., Optical and Electro-Optical Information Processing, chapter 33, pp. 591-614, Cambridge, Mass. : MIT Press, 1965.

46. R.J. SOLOMONOFF, 'A new method for discovering the grammar of phrase structure languages', Information Processing, Paris, 1960.

47. R.J. SOLOMONOFF, 'The mechanization of linguistic learning', Proceeding of the Second International Congress on Cybernatics, Namur, Belgium, 1962.

48. J.A. FELDMAN, 'First thought on grammatical inference', Stanford Artificial Intelligence Project Memo. no. 55, Stanford University, Stanford, California, August 1967.

49. J.A. FELDMAN and D. GRIES, 'Translator writing systems', Communications of the ACM, vol. 11, pp. 77-113, 1968.

50. P.R. ROSENBAUM, 'A grammar base question-answering procedure', Communications of the ACM, vol. 10, pp. 630-635, 1967.

51. T.G. EVANS, 'Grammatical inference techniques in pattern analysis', in J.T. Tou, Ed., Software Engineering, vol. 2, New York : Academic Press, 1971.

52. H. GENCHI, K.I. MORI, S. WATANABE and S. KATSURAGI, 'Recognition of handwritten numerical characters for automatic letter sorting', Proceedings of the IEEE, vol. 56, pp. 1292-1301, 1968.

53. S.K. CHANG, 'A method for the structural analysis of two-dimensional mathematical expressions', Information Sciences, vol. 2, pp. 253-272, 1970.

54. J.E. HOPCROFT and J.D. ULLMAN, Formal languages and their relation to automata, Reading, Mass. : Addison-Wesley, 1969.

55. A.V. AHO and J.D. ULLMAN, The Theory of Parsing, Translation and Compiling, vol.I : Parsing, Englewood Cliffs, N.J. : Prentice Hall, Inc., 1972.

56. N. CHOMSKY, 'Three models for the description of language', IRE Transactions on Information Theory, vol. IT-2, no. 3, pp. 113-124, September 1956.

57. N. CHOMSKY, 'On certain formal properties of grammars', Information and Control, vol. 2, pp. 137-167, 1959.

58. N. CHOMSKY, 'Formal properties of grammars', in R.D. Luce, R.R. Bush and E. Galanter, eds., Handbook of Mathematical Psychology, vol. II, New York : Wiley, 1963.

59. S. GINSBURG, The Mathematical Theory of Context-Free Languages, New York : McGraw-Hill, 1966.

60. S.C. KLEEN, 'Representation of events in nerve nets and finite automata', in Automata Studies, Princeton, N.J. : Princeton University Press, 1951.

61. E.M. GOLD, 'Language identification in the limit', Information and Control, vol. 10, pp. 447-474, 1967.

62. R. SOLOMONOFF, 'A formal theory of inductive inference', Information and Control, vol. 7, pp. 1-22, 224-254, 1964.

63. J.J. HORNING, 'A study of grammatical inference', Ph.D. dissertation, Department of Computer Science, Stanford University, 1969.

64. A.W. BIERMANN and J.A. FELDMAN, 'A survey of results in grammatical inference', in S. Watanabe, ed., Frontiers of Pattern Recognition, pp. 31-54, New York : Academic Press, 1972.

65. F.J. MARYANSKI, 'Inference of probabilistic grammars', Ph.D. dissertation, Department of Computer Science, University of Connecticut, 1974.

66. A.R. PATEL, 'Grammatical inference for probabilistic finite-state languages', Ph.D. dissertation, Department of Electrical Engineering, University of Connecticut, Storrs., 1972.

67. T.L. BOOTH, Sequential Machines and Automata Theory, New York : John Wiley and Sons, Inc., 1967.

68. F.J. MARYANSKI and T.L. BOOTH, 'Inference of finite-state probabilistic grammars', IEEE Transactions on Computers, vol. C-26, no. 6, pp. 521-536, June 1977.

69. I. FRIS, 'Grammars with partial ordering of the rules', Information and Control, vol. 12, pp. 415-425, 1968.

70. S. ABRAHAM, 'Some questions of phrase structure grammars', Computational Linguistics, vol. 4, pp. 61-70, 1965.

71. D.J. ROSENKRANTZ, 'Programmed grammars and classes of formal languages', Journal of the Association for Computing Machinery, vol. 16, pp. 107-131, 1969.

72. S.GINSBURG and E.H. SPANIER, 'Control sets on grammars', Mathematical Systems Theory, vol. 2, pp. 159-177, 1968.

73. R.A. THOMPSON, 'Determination of probabilistic grammars for functionally specified probability-measure languages', IEEE Transactions on Computers, vol. C-23, no. 6, pp. 603-614, June 1974.

74. R.A. THOMPSON, 'Language correction using probabilistic grammars', IEEE Transactions on Computers, vol. C-25, no. 3, pp. 275-286, March 1976.

75. K.S. FU, 'On syntactic pattern recognition and stochastic languages', in S. Watanabe, ed., Frontiers of Pattern Recognition, New York : Academic Press, 1972.

76. K.S. FU and T. HUANG, 'Stochastic grammars and languages', International Journal of Computing and Information Science, vol. 1, 1972.

77. K.S. FU, 'Stochastic languages for picture analysis', Computer Graphics and Image Processing, vol. 2, no. 3/4, December 1973.

78. D.V. HUNTSBERGER and P. BILLINGSLEY, Elements of Statistical Inference, Boston, Mass. : Allyn and Bacon, Inc., 1977.

79. V.I. LEVENSHTEIN, 'Binary codes capable of correcting deletions, insertions, and reversals', Soviet Physics-Doklady, vol. 10, no. 8, pp. 707-710, February 1966.

80. L.R. BAHL and F.JELINEK, 'Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition', IEEE Transactions on Information Theory, vol. IT-21, no. 4, pp. 404-411, July 1975.

81. V.M. VELICHKO and N.G. ZAGORUYKO, 'Automatic recognition of 200 words', International Journal of Man-Machine Studies, vol. 2, pp. 223-234, 1970.

82. R.A. WAGNER and M.J. FISCHER, 'The string-to-string correction problem', Journal of the Association for Computing Machinery, vol.21, no.1, pp. 168-173, January 1974.

83. D. SANKOFF, 'Matching sequences under deletion/insertion constraints', Proceedings of National Academic Science USA, vol. 69, no. 1, pp. 4-6, January 1972.

84. H. SAKOE and S. CHIBA, 'Dynamic programming algorithm optimization for spoken word recognition', IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-26, no. 1, pp. 43-49, February 1978.

85. R.E. BELLMAN, Dynamic Programing, Princeton, N.J. : Princeton University Press, 1957.

86. J. BACKUS, 'The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference', Proceedings of the International Conference on Information Processing, UNESCO, Paris, June 1959.

87. P.A. LYN, 'Economic linear-phase recursive digital filter', Electronics Letters, vol. 6, p. 143, 1970.

88. C.N. ALBERGA, 'String similarity and misspellings', Communications of the ACM, vol. 10, no. 5, pp. 302-313, May 1967.

89. S. KUNO and A.G. OETTINGER, 'Multiple-path syntactic analyser', in Information Processing 62, pp. 306-312, Amsterdam : North-Holland, 1963.

90. J. EARLEY, 'An efficient context-free parsing algorithm', Communications of the ACM, vol. 13, no. 2, pp. 94-102, February 1970.

91. G. LYON, 'Syntax-directed least-error analysis for context-free languages: a practical approach', Communications of the ACM, vol. 17, no. 1, pp. 3-14, January 1974.

92. D.G. HAYS, 'Language-data processing', in H. Borko, Ed., Computer Applications in the Behavioral Sciences, pp. 394-423, Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1962.

93. A.V. AHO and T.G. PETERSON, 'A minimum distance error-correcting parser for context-free languages', Siam Journal on Computing, vol. 1, no. 4, pp. 305-312, December 1972.

94. D.H. YOUNGER, 'Recognition of context-free languages in time n^3', Information and Control, vol. 10, pp. 189-208, 1967.

95. M.M. KHERTS, 'Entropy of languages generated by automata of context-free grammar with a single-valued deduction', Naucho-Tekhnicheskaia Informatsia, Ser. 2, no. 1, January 1968.

96. A.J. PUTMAN, 'Feature extraction for automatic recognition of telephone speech', Ph.D. Thesis, Department of Electrical and Electronic Engineering, University of Aston in Birmingham, England, to be submitted in 1979.

97. G. LEVI and F. SIROVICH, 'Structural description of fingerprint images', Information Sciences, vol. 4, pp. 327-356, 1972.

98. K.P. LI, G.W. HUGHES and T.B. SNOW, 'Segment classification in continuous speech', IEEE Transactions on Audio and Electroacoustics, vol. AU-21, pp. 50-57, 1973.