Computer assisted application of stochastic structuring techniques in musical composition and control of digital sound synthesis systems

by

Kevin Jones

A thesis submitted for the degree of Doctor of Philosophy
in the music section of the Centre for Arts and Related
Studies at the City University, London

June 1980

CONTENTS.

LIST OF FIGURES

ACKNOWLEDGEMENTS

My thanks go to my supervisors, both of whom have had to follow unfamiliar paths into foreign territory; to Dr. Malcolm Troup for his encouragement and musical vision, and to Dr. Jean Scott for her steady guidance and patient correction of technical details. I am also grateful to Stanley Haynes whose pioneering work in establishing computer sound synthesis facilities in this country greatly assisted my research in its initial stages, and whose invitation made it possible for me to work for a short time at IRCAM. I am also grateful to Jean Claude Risset for this opportunity, and to Gottfried Michael Koenig for the chance to work at the Institute of Sonology, Utrecht. My thanks are also due to David Jenkins who obtained funds for me to support these visits, and to the Worshipful Company of Musicians who sponsored the Electronic Music Studentship which I held whilst carrying out most of the research. I must also give grateful acknowledgement to Curt Roads who first drew my attention to the implications of the use of formal grammars and thus gave me a methodical framework in which to interpret and develop hitherto rather abstract ideas; also to Steve Holtzman who pointed out to me the equivalence of Markov Chains with finite-state grammars; and to Dr. Orin Smith who helped to guide me through some of the more tricky aspects of Markov Chain theory. I must thank the members of the New English Orchestra and the City University Singers who, with patience, perseverance and enthusiasm, have performed some of the works described in the thesis. Finally, I must express my gratitude to Dr. Graeme Bailey for his constructive interest, helpful discussion and practical assistance in typing and preparing the thesis.

DECLARATION

Powers of discretion are granted to the University Librarian to allow this thesis to be copied in whole or in part without further reference to the author. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

ABSTRACT.

Although many sophisticated computer sound synthesis systems have recently been developed, their use remains significantly under-exploited by composers. This is due to a large extent to the cumbersome and detailed programming invariably necessary in order to produce even the simplest of sounds. So, for effective and efficient use of computer sound synthesis techniques in musical composition, certain generative methods of forming and controlling data structures can be a powerful aid. This thesis proposes and develops such methods.

Techniques of pattern formation are described which have been developed and used in the original compositions of the author. These are based on stochastic generative schemes. The examples cited develop across the spectrum of complexity, ranging from the application of simple probability distributions through to intricate structures of inter-related stochastic constraints under the control of a general grammatical schema. In some cases the computer output has been designed so that it can be transcribed for performance by conventional musical forces. In other cases the output data has been used for direct control of digital sound synthesis programs and equipment. In addition to the limited practical resources existent at the City University, the author has been able to work with facilities made available on practical research visits to other institutions in Europe.

The structuring techniques employed by the author are compared with other research taking place elsewhere, and are set in the context of

recent advances in the development of certain formal grammars for the description of musical structures.

Methods of general applicatiǒn of the techniques are suggested, and in particular, in the light of the development of alternative methods of non-standard synthesis, the use of stochastic web grammars is proposed not only for the macro-structuring of notes or sound objects, but also in the definition of sound complexes and timbres. The techniques explored have an explicit use as a practical tool for composers and may be incorporated into, of form the basis of, computer controlled compositional aids of wide ranging application.

SYMBOLS AND ABBREVIATIONS

Chaper Two

IRCAM	Institute de Recherche et Coordination Acoustique/ Musique
VOSIM	VOice SIMulation

Chapter Four

E	event space
$\langle \; \rangle$	an ordered set of objects
$\langle e_1, e_2, \ldots e_n \rangle$	an ordered set of n events $e_1, e_2, \ldots, e_n$
e_i	event with index i (the i-th event in an ordered set)
$\|E\|$	order of the event space
E_{int}	intensity event space
E_{dur}	duration event space
E_{pitch}	pitch event space
P	probability assignment over an event space
$(\; , \; , \ldots)$	an ordered set of values
$p(e_i)$	the probability of event e_i occuring
$\leq$	less than or equal to
$\sum_{i=1}^{n} p(e_i)$	the sum of probabilities for events e_1 to e_n
i!	factorial of i : i (i-1) (i-2) ... 2 . 1

POD	POisson Distribution
rnd(n)	random function generating an integer between 1 and n
$\pi_1, \dots, \pi_n$	a sequence of random real numbers between 0 and 1
Π	an array consisting of random numbers $\pi_1, \dots, \pi_n$
DS	Difference Space
P_{DS}	probability assignment over a difference space

Chapter Five

RD function	random decaying function

Chapter Six

$p_{ij} = p(E_{m+1}=j \mid E_m=i)$	the probability of event e_j occuring next given that event e_i has just occured
P	stochastic matrix of probabilities p_{ij}
ALGOL 60	ALGOrithmic Language defined in the year 1960
$i \to j$	event j is accessible from event i
$\Rightarrow$	implies
$i \leftrightarrow j$	events i and j communicate
C	an equivalence class of events
$\in$	belongs to
$\notin$	does not belong to
$p_{ij}^{(n)}$	the probability that event i will be followed by event j after n steps
P^n	matrix of $p_{ij}^{(n)}$ probabilities

p_{ij}^*	the limiting value of $p_{ij}^{(n)}$ as n approaches infinity
P^*	matrix of p_{ij}^* probabilities
Π	the limiting distribution, an array equivalent to each row of P^*
π_j	the entries in Π $(\pi_j = p_{1j}^* = p_{2j}^* = \ldots = p_{nj}^*)$
P_C	probability matrix associated with equivalence class C
μ_j	mean recurrence time of event e_j
$>$	greater than
$\neq$	not equal to
M	sub-stochastic matrix associated with an alternating class
M_{ij}	sub-stochastic matrix in the i-th row and j-th column of a stochastic matrix lattice partitioned into alternating classes
ν_i	number of non-zero entries in the i-th row of a matrix
$\wedge$	and

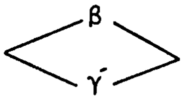
Chapter Seven

SAIL	Stanford Artificial Intelligence Language
GRAND	Generates RAND data file
GSTOK	Generates STOK data file
GRANP	Generates RANP data file
GRHY	Generates RHY data file
GFM	Generates FM data file
GGLIS	Generates GLIS data file
@	decimal exponent (standard in SAIL)
E	alternative decimal exponent (standard in FORTRAN)

FORTRAN	FORmula TRANslation language
Pi	the i-th item in a note definition
SEG	SEGments
SYNTH	SYNTHesize
V[ ]	event parameter vector
RAN(Ø)	a random real number between 0 and 1
FM	frequency modulation

Chapter Eight

$G = \langle V_N, V_T, P, S \rangle$	a grammar
V_N	a set of variables or non-terminals
V_T	a set of terminals
P	a set of production rules
$\rightarrow$	replace what is on the left of the arrow by what is on the right
S	starting symbol
$\cap$	set intersection
$\emptyset$	the null set
A	alphabet of a grammar
$\cup$	set union
A^*	the set of all possible strings generated by a grammar
L	the language generated by a grammar, the set of all permissable terminal strings
$\subseteq$	is contained in or equal to
V_T^*	the set of all possible terminal strings
{ , , ... }	an unordered set
α	any string from A^*

$\cup$	(reading down) contains
$/$	operator indicating a split in a different dimension
	web-diagram marker indicating that β and γ occur simultaneously
$/_n$	indexed operator indicating a split in the n-th dimension
$\mathcal{G}_S = \langle V_N, V_T, \mathcal{P}, \mathcal{S}, \mathcal{D} \rangle$	a stochastic grammar
$\mathcal{P}$	an ordered set of production rules
$\mathcal{D}$	a probability assignment over $\mathcal{P}$
M	fist-moment matrix
m_{ij}	entries of M. Each m_{ij} is equal to the sum of all the probabilities associated with each production rule where V_i is replaced by V_i multiplied by the number of times V_j occurs in the replacement string in each case
I	identity matrix
$\lambda_1, \ldots, \lambda_n$	eigenvalues or characteristic roots formed by setting the determinant of $M - \lambda I$ equal to zero
λ	principal eigenvalue
$\lvert \lambda \rvert$	absolute value of λ
$\emptyset$	the null string
NOT	NOTe

CHAPTER ONE

INTRODUCTION

1.1 Relation to other work

Very little material concerning the subject of computer music has been published in the United Kingdom. The Ph.D. thesis by Peter Manning of Durham University[1] is the most substantial contribution, which essentially contains a survey of the area and concentrates in particular on the hardware applications and related aspects of sound generation. There are a number of smaller articles of a general nature,[2] and others concerning applications in computer analysis,[3] but these are of little significance.

With the ready availability of large and sometimes even extravagant resources, a great amount of research has taken place in the United States which has generally been concerned with the establishment of digital sound synthesis systems. In particular, much effort has been directed into the analysis and computerized reproduction of natural instrument tones. Little work has been done to investigate compositional techniques using a computer. Some composers engaged in research, notably Xenakis, BrUn, Koenig and Truax, have made use of stochastic techniques in their work, but this has tended to be of a particular nature for limited tasks and does not approach the generality of the systems developed in this thesis. Markov Chains were suggested as appropriate models for music as early as the mid-fifties,[4] but even though the basic idea has been used by Xenakis and others, no other detailed development or analysis of the pattern structures which may be used has appeared of which the author is aware.

The author has already completed a thesis for a Master of Philosophy degree at the Computer Centre of the University of Aston in Birmingham entitled "A practical study of the application of Computer Techniques in processes of musical composition".[5] In addition to containing a fairly thorough survey of the historical situation, it is particularly concerned with various compositional applications of a certain group of limited techniques. The present thesis expands the field in the context of more general systems, examines the theoretical basis more rigorously, and extends into the application of stochastic structuring techniques to digital sound synthesis systems and in the definition of timbral complexes.

Descriptions of research in closely related areas have recently appeared by C. Roads of Berkeley, California, and S. Holtzman of the Computing Department at the University of Edinburgh. Their work is described briefly in two articles [6] but is expanded in more detail in two unpublished documents.[7,8] The systems they propose are grammatical structures for the description of music sound data. These are designed to be carefully defined by the composer and allow him to organize and control his ideas more efficiently. The function of these systems is rather different from that of the stochastic systems proposed in this thesis, which are designed to be self-generative structures, in which the composer is interested primarily in the end result and not necessarily in the processes which lead to it.

The distinction is between a *desciptive* environment in which the composer uses a formal grammatical framework in which to control and manipulate the structural definition of sound data combinations, and a *generative* environment in which the composer sets up parameter boundaries for the structures he wishes to generate, which may be vague or imprecise if that is expedient in the context of the composer's requirements, and in which situation he may be unaware of the actual grammatical model employed within the system.

Holtzmann, Berg and Koenig have described and experimented with techniques of non-standard synthesis which sometimes have elements of stochastic control involved.

At appropriate points in the thesis, reference is made to the research which has been cited above.

1.2 New findings and propositions

All of the compositional applications described in the thesis are unique.

The application of extended stochastic techniques to generate compositional structures and produce control data for digital sound synthesis systems is thought to be without similar precedent. There are instances (Koenig, Truax) where work of a related nature has been described. Again these are identified and compared in the thesis (see 3.3) and the distinction of the work of the author is made explicit.

The extended musical application of Markov Chains is considered for the first time in this thesis, where an original, intuitive approach to the mathematical theory has been developed, appropriate for this particular context. As a generalization extending from that, the application of Stochastic Web grammars is described, both for generating musical structures and for the creation of complex sound textures and musical timbres. This and a further extension into higher dimensions of structural definition and integration, is a completely original conception.

The practical suitability of Stochastic Web grammars as composing tools, or as part of a general composing system is suggested.

1.3 Sources, methods and techniques

In almost all cases, the author has intuitively developed structures and techniques to solve particular practical needs. Subsequently, reference has been made to the literature and amongst colleagues to establish links and comparisons with potentially similar research elsewhere; and further, the author has attempted to establish a sound theoretical basis for the work by investigating the mathematical models and theory of the structures which have been used. Reference has been made to standard texts on probability theory, stochastic processes, and formal grammars.

The author has had the opportunity to work at, and use the extensive practical computing and sound synthesis resources available at the Insitut de Recherche et Co-ordination Acoustique Musique in Paris during August 1977, and at the Institute of Sonology of the University of Utrecht in the Netherlands in August 1978. These occasions proved to be of great practical value, and the difference in the basic philosophy and approach to computer music of the two institutions was very stimulating. Participation in the UNESCO symposium on computer music at Aarhus, Denmark was also of considerable benefit.

Most of the ideas contained in the thesis have been worked out in actual musical compositions, which have nearly all been successfully performed.

1.4 Structure of the thesis

After a general breakdown of the basic components of compositional strategy in order to establish a background framework for description, the thesis proceeds to develop a new theory of stochastic processes in composition. Starting with a consideration of simple probability distributions it leads on to an analysis of various types of Markov Chain. In each case, having established the theoretical foundations, the structure of the compositions *FIRELAKE* and *MACRICISUM* are considered in detail, as examples of the techniques applied in practice.

Following these examples, the method is extended in a wider theoretical context, and leads into a consideration of stochastic web grammars. Experiments with these grammars are analysed and finally applications are described and assessed.

An appendix contains scores of some of the compositions considered, and listings of computer programs which have been developed and used in composition and experiment.

CHAPTER TWO

COMPOSING STRATEGY - A LINGUISTIC MODEL

It is useful in breaking down and analysing general compositional method, to pursue a linguistic analogy. Three components of creative linguistic activity can be isolated: the semantic, syntactic and phonetic.[1] Their basic relationship can be represented as in figure 1.

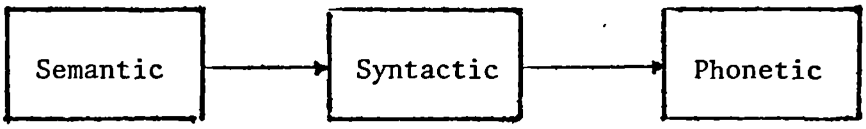


figure 1

A basic linguistic model of composing strategy

At the level of natural language, the *semantic* component refers to the mental activity of sorting out the meaning of what is to be said, the *syntactic* component refers to the process of mapping the meaning into the grammar and words of a commonly accepted language, and the *phonetic* component is the action of translating the words into audible speech and sound. Each of these activities is dependent on the others, and the boundaries of distinction are not necessarily well defined. Nevertheless it is a convenient and usable model of categorization.

This model has been developed in a musical context in the writings of Otto Laske.[2] Its use as a strategy in examining the compositional process is considered below.

2.1 Semantic component

In the context of musical composition, the semantic component covers the creative act of sorting out the initial ideas for a musical composition, developing an overall structural plan or deciding what is to be said.

For example, figure 25 (page 84) shows an overall structural scheme, which represents general patterns and shapes, and not absolute, final relationships, for the piece *FIRELAKE*. Deriving this initial, conceptual plan forms a part of the semantic aspect of the compositional activity.

Decisions made at a semantic level may also determine precisely what syntactic form is to be used or how it is to be used. In particular, in some of the types of computer music composition under consideration, semantic decisions may involve the actual definition of a syntactic structure, or a special adaptation of a given structure.

Again, this was the case with the same example, where a particular syntactic system (using the Random Decaying Function - see section 5.2) was defined.

It is therefore more accurate to adapt the figure above and re-draw it as in figure 2 below.

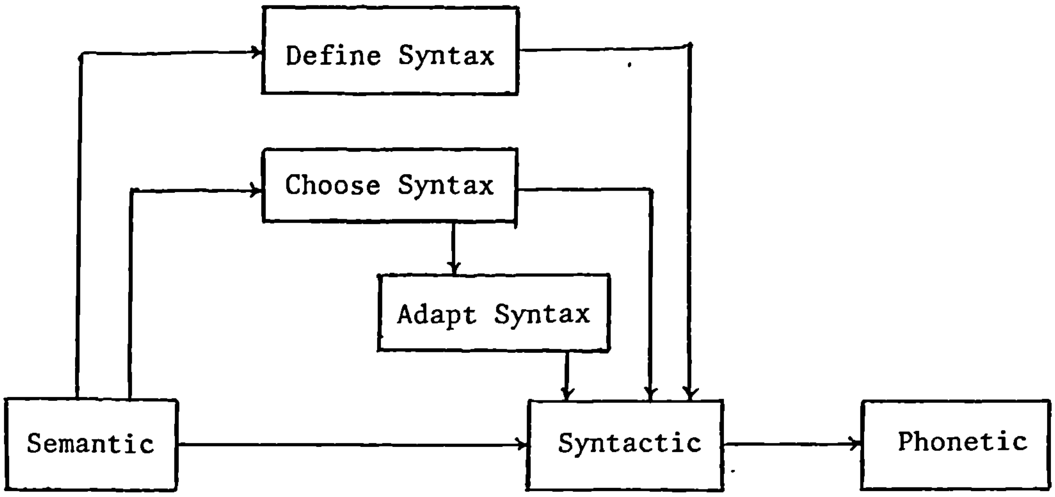


figure 2

An extended linguistic model of composing strategy

2.2 Syntactic component

So having determined what is to be said at the semantic level, the subsequent syntactic decisions determine how it is to be said, or what language is to be employed.

All composers have always worked within the framework of a familiar syntactic system, where an accepted language explicitly or implicitly defined by an agreed set of rules or grammar is employed. Modern composers have a wide range of syntactic systems from which to choose, which offer various degrees of rigidity. Such systems include conventional tonality, polytonality and modal systems, microtonality, serialism, total serialism which has very strict rules, and indeterminacy which has few. Very often composers have made use of their own specially defined syntactic systems, and even if not explicitly developed as such, distinctive composers' styles can be analysed in syntactic terms.[3]

The syntactic systems described and developed in this thesis can be used to model most of the composing styles listed above, and are consequently much more general in their scope.

It is the syntactic part of the composing process which is the main subject of this thesis, and its nature and application in computer music systems is considered in the next chapter and those following.

2.3 Phonetic / Sonic component

When considered in a musical context, the term phonetic is not really appropriate, and so the term *sonic* used by Laske[4] is more suitable.

The sonic component refers to the action of defining exactly which sorts of sounds are to be used, that is mapping the notes or symbols resulting from the syntactic activity into an audible form. At the level of conventional composition technique this is equivalent to the process of orchestration.

Again, the areas of activity as defined in the original diagram are rather blurred. The nature of the sonic mapping is frequently determined by decisions taken at the semantic level, and in computer music, systems can be defined using syntactic techniques. The final version of the diagram of composing strategy will thus emerge as in figure 3 (page 34).

In the past, most of the work in computer music research has been done at the sonic level. In order to communicate with the machine, the systems which have been developed have necessarily used basic syntactic structures for sound definition, but these are by their nature quite different to the generative grammars proposed later in the thesis.

The first widely used program for defining sound data was MUSIC V developed by Max Matthews at Bell Telephone Laboratories.[5] This uses as a model the modules of the traditional electronic music synthesiser which may be combined and built up in as complex an arrangement as desired. This program has spawned a host of related descendants which are the basis of most computer sound synthesis systems in use today. These

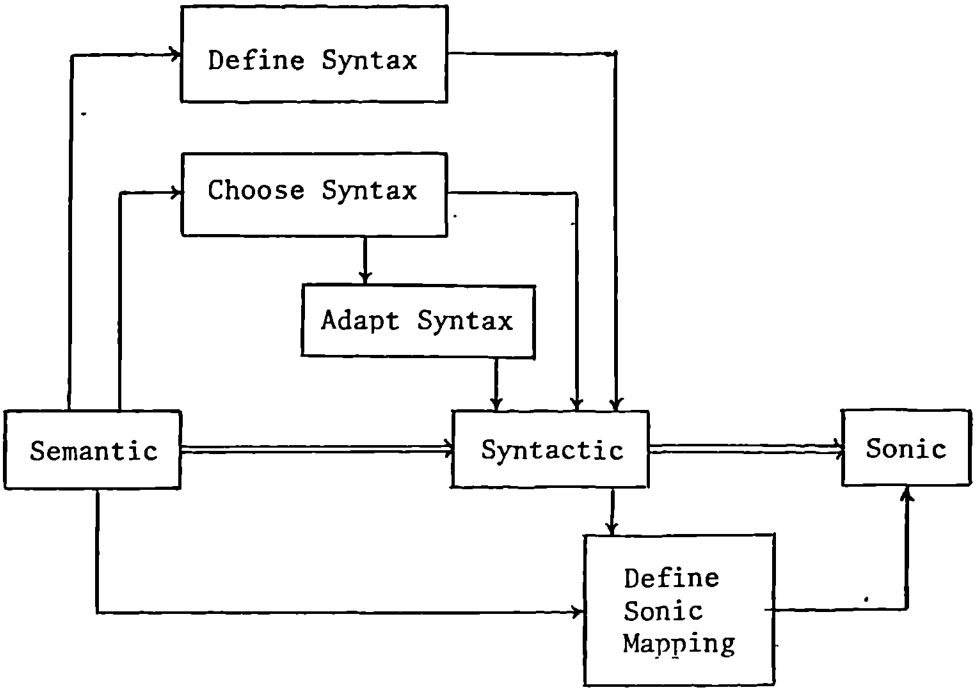


figure 3

A further extension of the linguistic model of composing strategy include MUSIC IVB, MUSIC IVBF[6], MUSIC 360, MUSIC XI , and MUSIC 10[7] which was used at IRCAM by the author to realise the composition *MACRICISUM* and is considered further in section 7.1.3 .

A popular technique which can be realised on such systems is frequency modulation. Its use as a powerful tool in computer sound synthesis was first seriously proposed and investigated by John Chowning of Stanford University[8] and has proved to be so successful that many hardware systems have been designed snecifically for FM generation. The technique was used in *MACRICISUM* and is described in section 6.5.2 .

Another alternative source of interesting sounds is supplied by the VOSIM generator developed by Kaegi at Utrecht.[9] This was used to create the taped sounds for *DUTCH COURAGE* realised by the author at Utrecht.

All of the above techniques are based on traditional approaches to sound synthesis which are essentially built upon Fourier theory. Certain alternative methods of synthesis have recently been proposed and are usually referred to as "non-standard". Most of this work has emerged from research taking place at Utrecht by Berg, Koenig and Holtzmann. "Standard" synthesis is based on top-down techniques where an acoustic model is used to derive samples defining a sound pressure wave. By contrast, the "bottom-up" techniques of non-standard synthesis specify sound samnles by some syntactic method without reference to traditional acoustic theory.

In each of these programs, an abstract pattern of numbers, related only to each other, is specified by the control program. Van Prooijen's CYCLE[10] program is basically geared towards making small incremental changes as a basic initial number pattern is repeated a large number of times. Berg's PILE 2 [11] describes a fairly sophisticated syntax for defining and manipulating lists of numbers. Koenig's SSP [12] has a simpler structure which includes options for stochastically defining sequences and repeating groups of values. SSP is very similar to an earlier program of Koenig's: PROJECT 2, which was developed for defining compositional structures at note level. Holtzmann's "automated digital sound synthesis instrument" [13] uses a generative grammar to define sequences of machine instructions. Brun's SAWDUST [14] program, as well as having basic block manipulation functions, has provision for using polynomials to define strings of values corresponding to a curve linking specific start and end points.

The advantage of these approaches is that they are not bound by the limitations of an imperfect acoustic theory, and some rather extraordinary, hitherto unimaginable sounds have been generated in this way. Many of these sounds, however, are so strange that to most ears they appear harsh and unintelligible. Nevertheless these methods do have a significant potential for development.

The author's own work has used generative techniques at the sonic level, to control data expressed by "standard" synthesis methods, as for example in the second movement of *MACRICISUM* , entitled *LAITRAPARTIAL* , which is described in section 6.3 below.

CHAPTER THREE

GENERATIVE SYSTEMS OF SYNTACTIC CONTROL

3.1 The need for generative systems

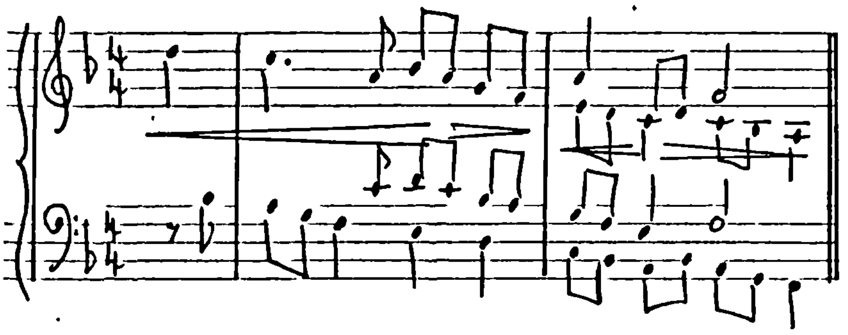
The computer sound synthesis systems in general use are almost all based on the model set up by Max Matthew's MUSIC 5 program. They require large amounts of data to first of all define "instruments" and then to specify lists of notes which will be "played" on the instruments. Example 3.1 (page 38) lists the MUSIC 5 data necessary to define an instrument and play the notes of the short musical fragment shown.

Now this is a rather trivial example which will merely sound like a rather poor electronic organ, for even with such large amounts of data, relatively static and unchanging sound quality will be oroduced. Various techniques such as frequency modulation are possible to produce interesting time-variant timbres which escape from an artificially consistent "electronic" sound quality, but even then large lists of notes are still required.

Max Matthews, recognising this fact, made provision for the MUSIC 5 user to write what he terms PLF routines [1] which can perform elementary operations on groups of notes. But this facility is essentially just a method for repeating notes in various ways. Other formal systems for note organization have been used and proposed, and these are considered below (3.4).

figure 4

A musical fragment with the MUSIC 5 data to define it



```
SV2 0 3;COM NO CONVT;
SI3 0 10;COM DISC OUTPUT WITH REPORT;
SIA 0 4 10000; COM SAMPLING RATE 10000;
COM INSTRUMENT DEFINITION;
INS 0 1;
OSC P5 P6 B2 F3 P30;
OSC B2 P7 B2 F6 P29;
OUT B2,B1;
END;
INS 0 2;
OSC P5 P6 B2 F4 P28;
OSC B2 P7 B2 F6 P27;
OUT B2 B1;
END;
INS 0 3;
OSC P5 P6 B2 F5 P26;
OSC B2 P7 B2 F6 P25;
OUT B2 B1;
END;
GEN 0 1 3 0 0 .99 20 .99 491 0 511;
GEN 0 1 4 0 0 .99 20 0 511;
GEN 0 1 5 0 0 .99 20 .75 40 .75 491 0 511;
GEN 0 1 6 0 0 .99 50 .99 205 -.99 306 -.99 461 0 511;

COM SOPRANO;

NOT 0    1  1     750  0.0511  30.047;
NOT 1    1  1.5  1259  0.0341  26.776;
NOT 2.5  1  0.5  1750  0.1022  22.484;
NOT 3.0  1  0.5  2000  0.1022  23.813;
NOT 3.5  1  0.5  1750  0.1022  22.484;
NOT 4.0  1  0.5  1500  0.1022  20.031;
NOT 4.5  1  0.5  1250  0.1022  17.834;
NOT 5    1  1    1000  0.0341  22.484;
NOT 6    1  0.5  2000  0.1022  13.388;
NOT 6.5  1  0.5  1500  0.1022  15.023;
NOT 7    1  2    1000  0.0256  17.834;
```

```
COM ALTO;

NOT 5     2  0.5  1000  0.1022  16.863;
NOT 5.5   2  0.5  1500  0.1022  15.023;
NOT 6     2  1    2000  0.0511  13.388;
NOT 7     2  0.5  1500  0.1022  13.388;
NOT 7.5   2  0.5  1000  0.1022  11.906;
NOT 8     2  1     750  0.0511  11.242;

COM TENOR;

NOT 2.5   2  0.5  1750  0.1022  13.388;
NOT 3     2  0.5  2000  0.1022  15.023;
NOT 3.5   2  0.5  1750  0.1022  13.388;
NOT 4     2  0.5  1500  0.1022  11.906;
NOT 4.5   2  0.5  1250  0.1022  11.242;
NOT 5     2  0.5  1000  0.1022  10.007;
NOT 5.5   2  0.5  1500  0.1022   8.917;
NOT 6     2  1    1750  0.0511   8.432;
NOT 7     2  2    1000  0.0256   8.917;

COM BASS;

NOT 0.5   3  0.5  1000  0.1022  11.906;
NOT 1     3  0.5  1250  0.1022  11.242;
NOT 1.5   3  0.5  1500  0.1022  10.007;
NOT 2     3  1    1750  0.0511   8.917;
NOT 3     3  1    2000  0.0511   8.432;
NOT 4     3  1    1500  0.0511   7.512;
NOT 5     3  0.5  1000  0.1022   6.694;
NOT 5.5   3  0.5  1500  0.1022   5.953;
NOT 6     3  0.5  2000  0.1022   5.621;
NOT 6.5   3  0.5  1500  0.1022   5.953;
NOT 7     3  0.5  1250  0.1022   5.621;
NOT 7.5   3  0.5  1000  0.1022   5.003;
NOT 8     3  1     750  0.0511   4.458;
TER 9;
```

figure 4 continued

Techniques which can merely repeat groups of notes are not very powerful, and in order to build complex and interesting structures, it is necessary to find a method of generating large amounts of variable data.

Many composers have tended to shy away from computer music systems. Those who have been bold enough to tackle the difficulties and have taken the trouble to learn how to use the systems, have discovered that it takes a great amount of time and energy to produce even the simplest of sound structures. It is possible to spend days preparing lists of data to define sounds lasting a few seconds only. Many of those composers who have made use of computers have barely scratched the surface of the rich resources available, and have merely used the computer to reproduce compositions essentially instrumental in conception and in a strongly note-based idiom. There is no reason why systems should not be used in that way, when advantage may be taken of very accurate capabilities for control of pitch, rhythm and intensity which instrumental players are unable to achieve, but the possibilities which computer music generation have to offer are far richer and more interesting than such a restricted use might suggest.

At the recent UNESCO symposium on computer music at Aarhus, it was generally agreed that much of the current usage of computer music systems was trivial and that there was an urgent need at the present time to encourage more composers to make use of the available systems and to develop compositional techniques appropriate to the resources.

In the United States in particular, the development of hardware and sound synthesis software has reached a sophisticated level, but the sometimes extravagant systems are not being fully exploited.

Unfortunately, such a favourable situation with respect to resources does not exist within the United Kingdom!

Generative systems, then, have much to offer in the field of computer music composition. In addition to helping to define large, interesting and varied data structures, generative techniques are able to spur the imagination of the composer, and may sometimes suggest possibilities which would not otherwise have been anticipated. Furthermore, such systems can be intrinsically valuable methods of organising sound structures in their own right.

William Buxton has made a distinction between "composing programs" and "computer-aided composition".[2] Koenig has pointed out in a recent lecture that such a distinction merely identifies two extremes, and they do not form distinct entities.[3] There are programs and systems which fall between the two, and there are some which may be both at once! Ideally, a program which can function as the most general composing aid should offer a full range of possibilities from complete user control to almost total control by the program. The structures under consideration can be used at any point on this scale.

Although the main purpose of this thesis is to investigate stochastic processes, some other alternatives will be considered first of all.

3.2 Deterministic Processes

The simplest type of generative system, or composing routine is a deterministic process in which the composer defines a rule which can be repeatedly applied to a set of notes. Once the process is begun it may continue to run indefinitely. Such processes usually involve defining a basic pattern of some sort and then repeating it, but making incremental changes on each repetition.

Example 3.1 A 'musical chairs' process.

A short rhythmic fragment is defined (see figure 5(a), page 43). A rhythmic sequence can be generated by repeating the fragment, but removing the last note or rest on each repetition (figure 5(b)).

Example 3.2 An expansion process.

A group of pitches is defined (see fipure 6(a), page 43). A pitch sequence can be generated by repeating the group, and pushing each note up or down a semitone either side of a tonal centre with each repetition (figure 6(b)).

The above examples may be combined to produce the result of figure 7. Notice that the base fragments of examples 3.1 and 3.2 are of different lengths. By repeatedly using the same rhythmic pattern, and expanding the pitch pattern further and also applying it repeatedly, a sequence may be produced which will continue for a long time before it begins to repeat itself.

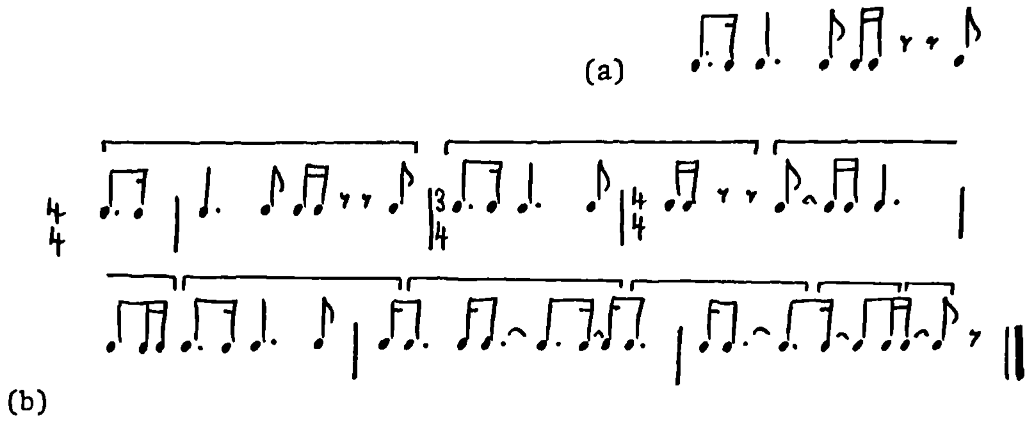


figure 5

A contracting rhythmic process

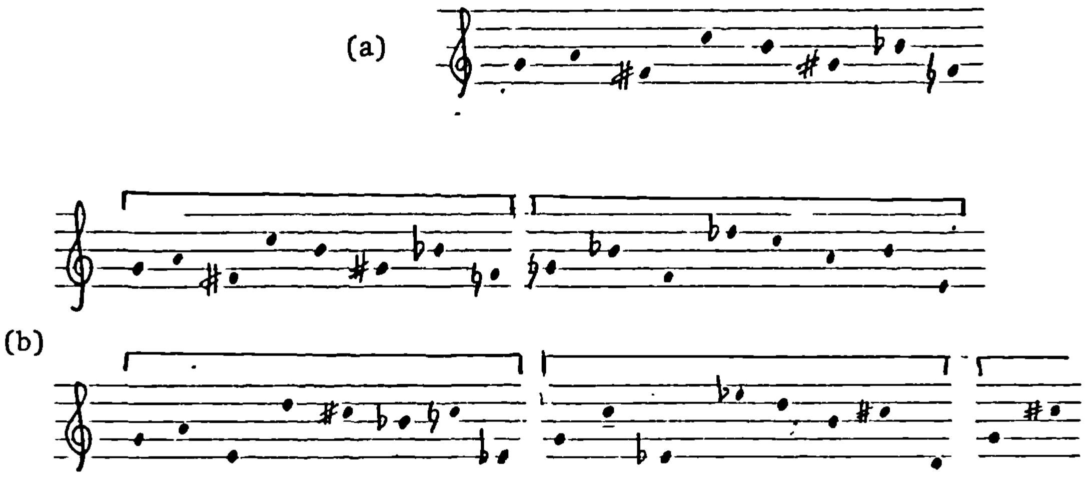


figure 6

A pitch expansion process

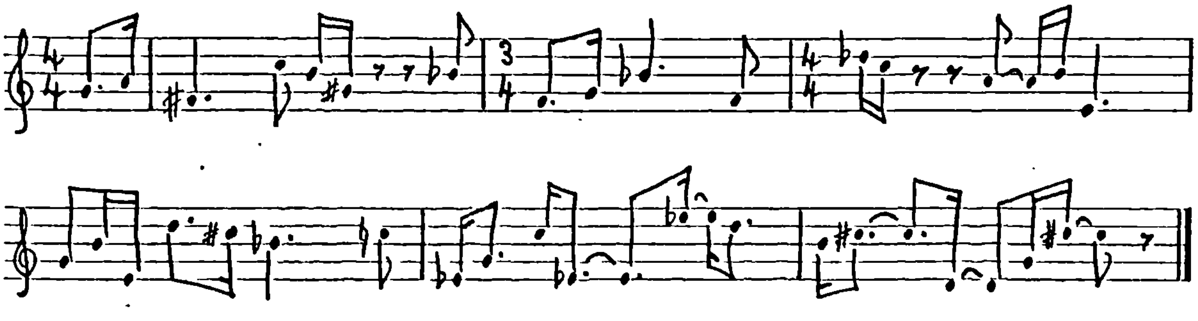


figure 7

A contracting rhythmic process and pitch expansion process combined

Another type of generative process is the system of change ringing, where slight alterations are made in the order in which a group of bells is rung on each repetition.

Example 3.3

On a group of six bells, the rule that the central pair changes order, then the two middle pairs, then the outer pairs, with the process repeated indefinitely, will produce the following sequence:

1	2	3	4	5	6
1	2	4	3	5	6
1	4	2	5	3	6
4	1	2	5	6	3
4	1	5	2	6	3
4	5	1	6	2	3
5	4	1	6	3	2
5	4	6	1	3	2
5	6	4	3	1	2
6	5	4	3	2	1
6	5	3	4	2	1
6	3	5	2	4	1
3	6	5	2	1	4
3	6	2	5	1	4
•	•	•	•	•	•

etc.

Some beautiful symmetries emerge in these patterns, which have inspired a number of mathematical studies of change ringing and may be modelled using group theory techniques. At least one of these studies has been carried out with the aid of a computer.[4] Change ringing processes have been used to generate material in some of Lejaren Hiller's computer music compositions.[5]

Generative processes of the type described have been used by many recent composers. The music of Steve Reich is almost entirely dedicated to exploring and making use of such processes.[6] But other composers have made effective use of process techniques in more familiar concert works. Messiaen, for example, has made use of an intervallic expansion process similar to that described in example 3.2 in many of his works.[7]

Although some composers have found that such deterministic composing processes can be very useful in specific instances, in general they are generatively rather weak structures compared to the stochastic systems to be considered below, which are of much wider application and offer more sophisticated opportunities for structural control.

3.3 Serial manipulation

In order to assist with data generation in many early attempts at composition with computer music systems, use was made of serial composition techniques, by means of which, the standard permutations and transpositions of a twelve-note series could be systematically exploited.

An early use in England was made in 1963 by the computer scientist Stanley Gill in response to a request for background music for a T.V. program.[8] He used a computer to generate versions of a twelve note row, which were tested against a few other compositional rules, and used to produce a short piece for string trio.

At Utrecht, Koenig has developed a composing program PROJECT ONE which is designed as an extension of serial techniques:

> "PROJECT ONE is intended to carry on from serial compositional technique, and to describe a study model which, practically independently, produces any number of variants of a basic structure." [9]

This was designed to produce an output which could be transcribed for whatever instrumental resources the user required. Koenig has recently renewed the program so that it can be used to generate electronic sounds directly. The author has made use of the nrogram in this form.

The greatest use of serial techniques, however, has been in the United States. The post-serial combinatorial techniques of composition favoured by Milton Babbit and his pupils have found a ready and convenient outlet in computer sound synthesis systems. Barry Vercoe[10] and John Melby[11] are amongst those composers who have used systems in this way.

A twelve-note serial facility is not uncommon in computer music installations in the States. For example, Donald Byrd has described the program MUSC which is available at Indiana.[12] A number of studies have appeared concerned with combinatorial problems which may arise in the systematic application of serial techniques. An article by Kobin and Ashford, for example, discusses certain harmonic difficulties which have been encountered.[13]

Some composers find that working with such techniques can be satisfying and rewarding, but nevertheless, the writer is of the opinion that again, they are rather limiting and lacking in generality. In fact Koenig points out that serial composition can be considered to be a special case of aleatoric compositional technique where the twelve notes have to occur with equal frequency.[14] Koenig's intention in using the term "aleatoric" is essentially to mean the same as the more explicit term "stochastic". Such systems are considered below (section 3.5).

3.4 Descriptive grammars

So far, the systems described have all been *generative* systems where a starting structure and a simple incremental or permutational algorithm are defined and used to produce a completely controllable and predictable result. In addition to these techniques, more formal *descriptive* systems have been developed which give the user of computer music synthesis systems more flexibility in organising sound data. These range from basic methods for automatically repeating chunks of data, which avoid unnecessary re-writing of lists of note definitions, to sophisticated data control systems where groups of related structures may be defined and various structural manipulations applied. Although not necessarily explicitly described as such, all these approaches can be termed types of descriptive grammars.

Examples of these are the SCORE program of Leland Smith, in use at Stanford and IRCAM,[15] and the facilities available as part of the Structured Sound Synthesis Project (SSSP) under development by William Buxton and others at Toronto.[16]

Recently, two sophisticated studies of a musical descriptive grammar have appeared which, based on the effective use of formal language theory, appear to offer the most powerful and efficient means of describing musical structures so far developed, though both have yet to be applied in a working system.[17, 18] It has already been stated that this study is more concerned with generative, rather than descriptive systems; but formal grammatical techniques, applied in the context of the auto-generative stochastic systems being considered, rather than exclusively as a medium in which to express a precise structural definition, have an enormous potential. Such a use of formal grammars is taken up and explored in chapter 8 and in the remainder of the thesis.

3.5 Stochastic processes

A stochastic process enables the user to define a precise set of limits, within which absolute control of activity is not specified. The results of applying a stochastic process will always be consistent and lead to the outcome desired by the user, although the actual details will be different. The stochastic process should be defined and applied in such a way that the actual nature and distribution of the details, which will be chosen at random, are not important, and that the significant structural aspects are carefully controlled.

A distinction should be drawn between a stochastic process and an *aleatoric* process. An aleatoric process (from the Latin *alea* : dice) implies that all outcomes are equally possible, there is no patterning process involved. This is what most people would understand to be meant by a *random* process. A stochastic process, on the other hand, produces results whose distribution and structure are perfectly controlled and predictable in general terms, and only the actual details (how detailed is up to the user) are unpredictable. Stochastic processes are sometimes termed 'random processes' by a few mathematicians, but this term can be misleading for the reason given above.

Stochastic processes are effective generative systems for use in computer music composition. Very often a composer wishes only to specify general structural tendencies for aspects of a composition, and may not be concerned with absolute details. Even in conventional composition, the composer does not specify exactly what sound is to be produced. He relies on conventions, the performer's interpretation, and completely random or uncontrollable influences on the final heard result. In using a computer, a composer is forced to specify every last aspect of a sound, and may unexpectedly find that he needs to control all sorts of parameters which were previously taken for granted. If these parameters are left unchanged, the result can be boring and lifeless; if an attempt is made to control every last subtlety of sound production individually, the task becomes hopeless; stochastic techniques can provide useful solutions to such difficulties at every level of composition.

Pioneering work with some types of stochastic control has been a feature of some of the early compositions of Xenakis [19] and Hiller.[20] Koenig has written composing programs involving stochastic choices, where the user is able to define an outline of a compositional structure for which the computer will fill in the details.[21] Some of Koenig's techniques have been developed and incorporated in Truax's P.O.D. programs.[22]

The purpose of the following sections is to explain the nature and operation of stochastic techniques in composition. The text begins by describing simple structures, and with the aid of examples builds up to a more complex and sophisticated level where the benefits of stochastic and grammatical means of definition are combined.

CHAPTER FOUR

STOCHASTIC PATTERN GENERATION (I)
SIMPLE PROBABILITY DISTRIBUTIONS

4.1 Preliminaries

Before using any sort of stochastic patterning system, it is necessary to identify which compositional parameters are to be controlled, and the range of values which each can assume. The range of values will be referred to as the *set of events* which constitute the *event space*. For most musical purposes the set of events will be discrete. The stochastic modelling of processes with continuous event spaces is rather more complicated; the present discussion will in general be limited to discrete event spaces.

The event space must be an ordered space, that is with the events in some natural or clearly defined sequence. The following notation will be used:

$$E = \left\langle e_1, e_2, e_3, \ldots, e_i, \ldots, e_n \right\rangle$$

where E is the event space, the ordered set of events
and $e_1, e_2, \ldots$ are the individual events
and i is the *index* of event e_i .

$|E|$ is the *order* of the event space and is equal to n , the number of events.

Example 4.1.1

An event space can be defined for the intensity parameter as below:

$$E_{int} = \left\langle \text{pp , pp , p , mp , mf , f , ff , fff} \right\rangle$$

here, e_1 = ppp , e_4 = mp , etc. $|E_{int}| = 8$

Example 4.1.2

An event space for the duration parameter may be:

E_{dur} = ⟨ 𝅘𝅥𝅯 , 𝅘𝅥𝅮 , 𝅘𝅥 ⟩

and in this case, $|E_{dur}| = 3$

It is not necessary to define an event space by listing its elements, a description will suffice.

Example 4.1.3

An event space for the pitch parameter:

$$E_{pitch} = \left\langle \text{the major scale of C} \right\rangle$$

here, $|E_{pitch}| = 7$

The techniques discussed below set out methods of selecting events to achieve a desired result.

4.2 Equally likely outcomes

The simplest situation is where each event is equally likely to occur. This corresponds to the aleatoric process described above where the outcome is totally at random.

Example

The following four parameter event spaces may be defined:

$E_{pitch} = \langle$ the scale of chromatic notes from C to G' $\rangle$

$E_{dur} = \langle$ 𝅘𝅥𝅯 to 𝅝 in semiquavers $\rangle$

$E_{int} = \langle$ ppp to fff $\rangle$

$E_{timbre} = \langle$ flute , oboe , violin $\rangle$

$|E_{pitch}| = 20 \qquad |E_{dur}| = 16$

$|E_{int}| = 8 \qquad |E_{timbre}| = 3$

A random choice is made from each parameter space. So, for example, the following matrix identifies an event by its index, for each parameter, in defining each note:

	pitch	dur.	int.	timbre
note 1	9	2	8	3
note 2	16	10	2	1
note 3	2	2	5	3
note 4	3	3	1	3
note 5	19	7	1	2
note 6	17	11	7	1

Or in terms of actual events:

note 1	A	♪	fff	violin
note 2	E	𝅗𝅥⁀♪	pp	flute
note 3	D	♪	mf	violin
note 4	D	♪.	ppp	violin
note 5	F#	♩..	ppp	oboe
note 6	E	𝅗𝅥⁀♪.	ff	flute

When transcribed into traditional musical notation, this will appear as in figure 8.

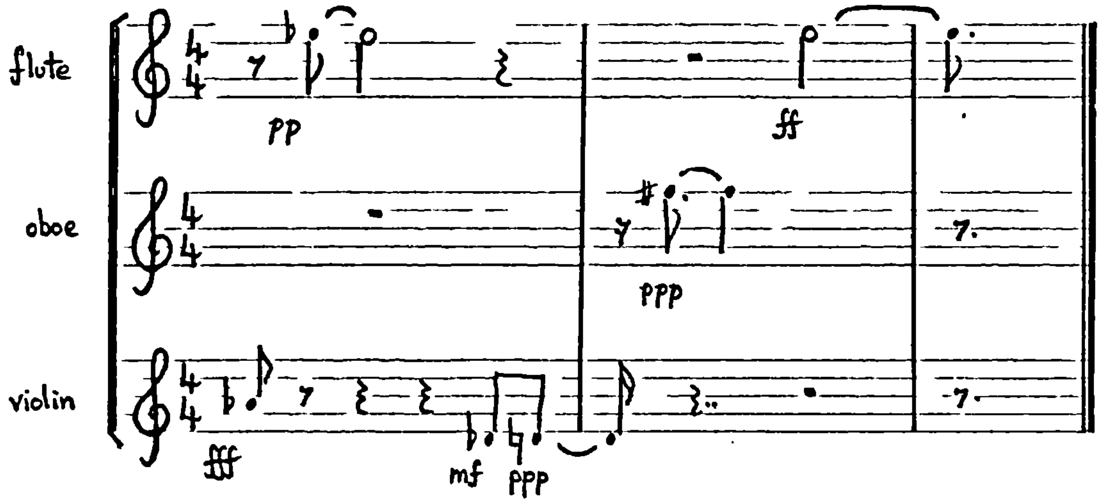


figure 8

A randomly generated sequence

It is a simple matter to use the random number generating facility available in most computer languages to control the choice of states. In the BASIC language, a call of the function RND(N) will deliver a random integer in the range 0 to N-1 , which can be used to choose the index of an event to be selected from the event space. Where a random number generator is not available, a simple subroutine or procedure may be written. The author has described a technique used in some earlier work, and there are many descriptions and discussions of random number generators in the literature.[1]

The material used at the opening of the tape piece *MACRICISUM* was generated by making a simple random choice from within the limits of the event space specified for each parameter (see section 6.2.3).

With many parameters being controlled, and with careful definition of each event space, the resulting sound, over short periods, will sound far from chaotic, and may produce pleasing and original patterned sequences.

Such a strict random distribution of events, with equally likely outcomes does not, however, correspond to any natural phenomena. It is usual for certain events to be weighted so that they are more likely to occur than others. Techniques for adjusting the likelihood of particular outcomes can be very useful in stochastic composition, and these will now be considered.

4.3 Probability distribution defined *a priori*

The weighting of various events can be specified by defining a *probability assignment* or *vector* over the event space. The number of entries, or order of the probability assignment must obviously equal the order of the event space.

Example 4.3.1

A composer could decide that he wants to produce a sequence of pitches which contains many A's, a few D's, hardly any F#'s, G#'s or B's, and nothing of anything else. He will first of all define a pitch event space:

$$E_{pitch} = \left\langle A , B , D , F\# , G\# \right\rangle$$

and a corresponding probability assignment:

$$P_1 = (0{\cdot}6 , 0{\cdot}05 , 0{\cdot}25 , 0{\cdot}05 , 0{\cdot}05)$$

or if listed side by side:

A	0•6
B	0•05
D	0•25
F#	0•05
G#	0•05

$p(e_i)$ is the probability of event i occuring, so

$$p(e_i) = p(A) = 0{\cdot}6$$

This means that A will occur with a probability of 0•6 , that is 60% of the time; B will occur with a probability of 0•05 , that is 5% of the time, and so on.

Each event nrobability must be less than or equal to one: $p(e_i) \leq 1$ for all i. The total sum of probabilities must equal one, or in the usual mathematical notation:

$$\sum_{i=1}^{n} p(e_i) = 1$$

This distribution can then be used to generate a sequence which stays within the required bounds.

A probability assignment may be used in a computer program to generate a sequence in the following way. First of all, starting at the beginning of the assignment and working through it, each probability is replaced by the sum of the probabilities up to that point, thus producing a cumulative array. So if $p'(e_i)$ represents the value associated with event e_i in the new cumulative array P', then:

$$p'(e_i) \longleftarrow \sum_{j=1}^{n} p(e_j) \qquad i = 1, \ldots, n$$

For each event required, a random real number between 0 and 1 is generated. The values in the array are compared one by one with the random number until a value greater that the random number is found. The position in the array where this occurs is used to index the required event from the event space. This procedure may be repeated for as many events as are required and will, as the total number of events generated increases, produce a sequence whose overall distribution of events becomes progressively closer to that defined in the original assignment.

Example 4.3.2

The probability distribution P_1 of example 4.3.1 above will become:

$$P_1' = (0\cdot6 , 0\cdot65 , 0\cdot9 , 0\cdot95 , 1)$$

The random number sequence:

$$\cdot34 \quad \cdot12 \quad \cdot71 \quad \cdot63 \quad \cdot88 \quad \ldots.$$

will produce the event sequence:

$$e_1 \quad e_1 \quad e_3 \quad e_2 \quad e_3 \quad \ldots. \quad \text{or} \quad A \quad A \quad D \quad B \quad D \quad \ldots.$$

The following sequence represents a typical pattern that would result from continuing such a generation:

A A D B D A A F# A A A D A A A D A A D G# A

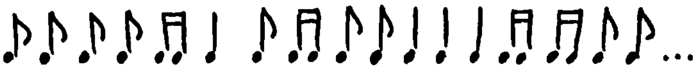
In musical terms, it can be seen that a distribution such as the one in 4.3.1 forms a very simple type of harmonic system, where a basic keynote, "A" in this case, predominates, and certain favoured note groupings and implied chords result.

Another event space and probability assignment could be defined for generating rhythmic patterns:

E_{rhythm} = ⟨ ♫ , ♪ , ♩ ⟩

$$P_2 = (0\cdot25 , 0\cdot5 , 0\cdot25)$$

which might generate a sequence such as the followinp:



When these sequences for pitch and rhythm are combined, the resulting sequence of figure 9 is formed:

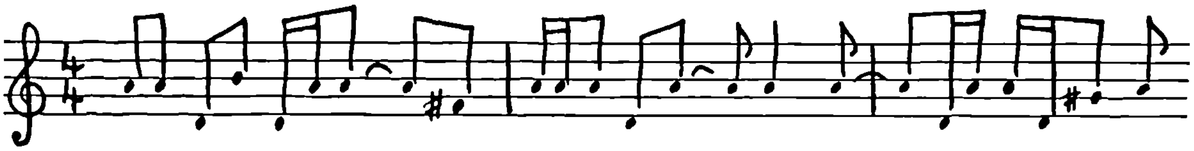
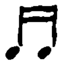


figure 9

A melodic sequence generated from simple probability distributions

Note that there is no need for events in the event space to consist of single elements. In the rhythmic event space of the example above:

e_1 = 

Each event may be whatever the composer wants to specify, and may be as complex as he wishes. A set of pre-recorded concrete sounds may be used for an event space, or a group of electronically synthesized sound complexes. The following simple example has each event in the event space consisting of a pre-composed group of notes.

Example 4.3.3

Using the same probability measure as above (4.3.1) a different event space of order 5 may be chosen with the five events $e_1,\ldots,e_5$ defined in figure 10 (page 61). An event sequence such as the following:

$$e_1\ e_1\ e_5\ e_1\ e_3\ e_2\ e_1\ e_3\ e_1\ e_1\ e_1\ e_5\ e_3\ e_1\ e_4\ e_1\ e_1\ e_3\ e_1\ e_1$$

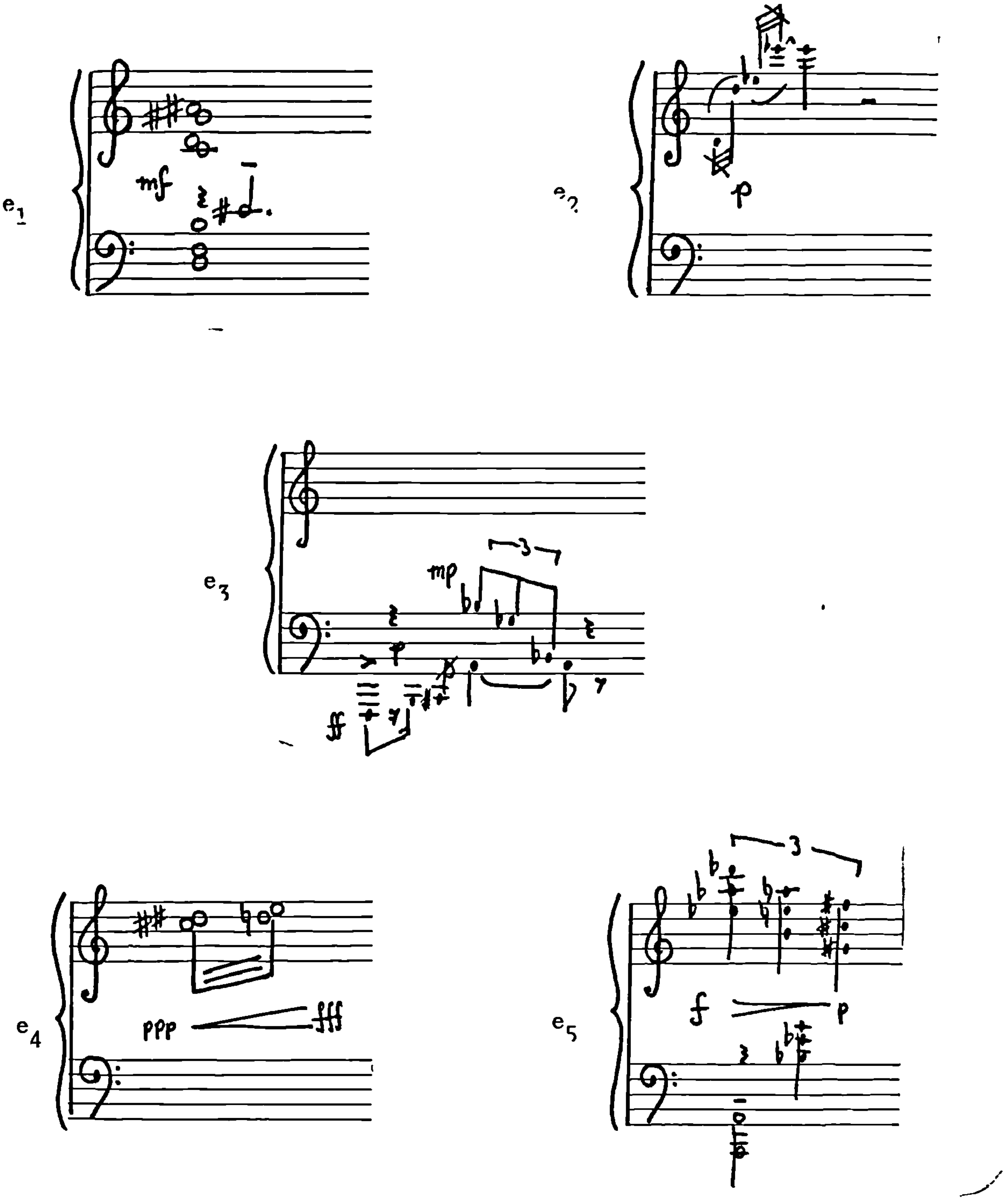


figure 10

An event space for a simple piano composition will then transcribe into the form of figure 11 (page 63).

It is possible for each event to be itself defined stochastically. This idea is taken up later (section 6.5.2) and is the case with many of the examples in chapter 7. Such generality gives this stochastic approach a considerable power.

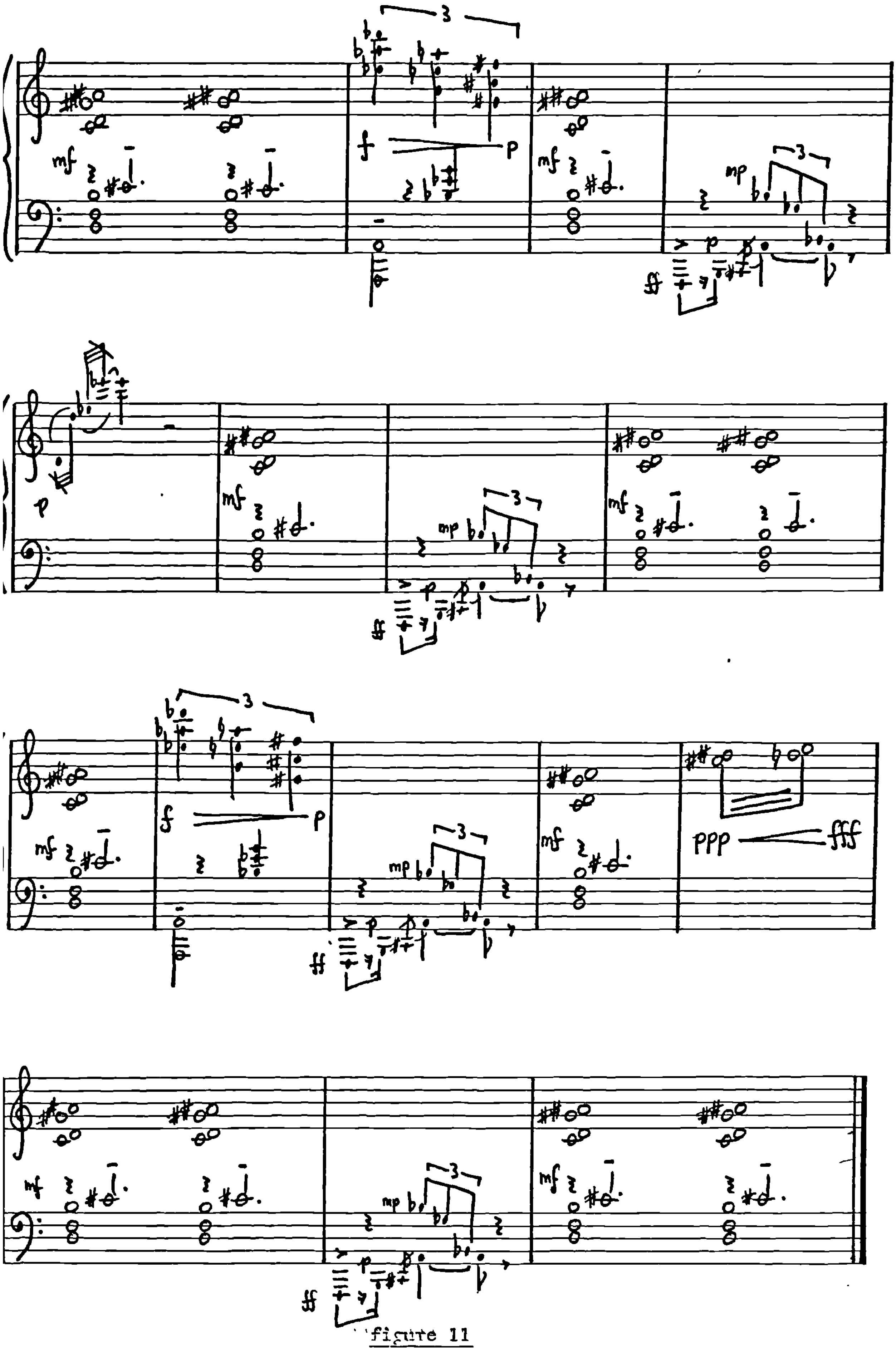


figure 11

An event sequence formed by the application of a simple probability assignment over the event space of figure 10

4.4 Probability distribution defined by analysis

It is possible to produce a probability assignment by analysing a given sequence. This is done by counting the number of occurences of each event and assigning probabilities proportionately.

Example 4.4.1

A probability assignment will be constructed for the extract shown in figure 12 from the first Chromatic Invention in Bartok's Microkosmos Book VI.[2]

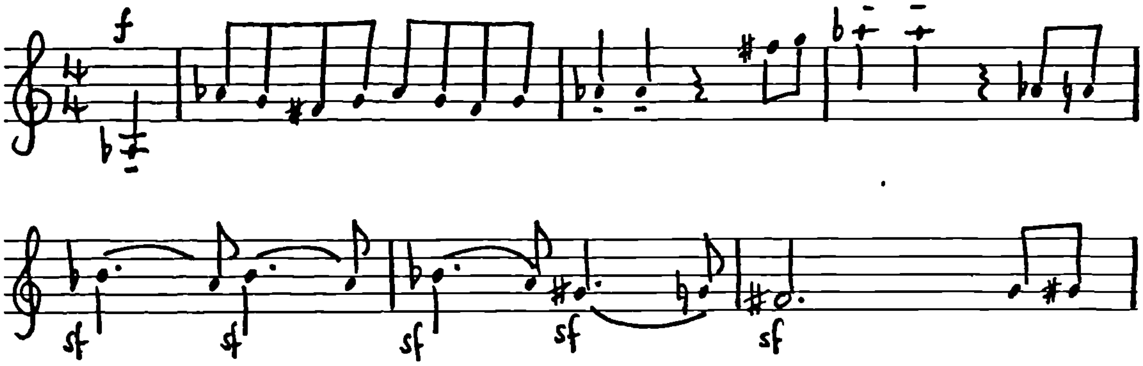


figure 12

The beginning of the First Chromatic Invention from Mikrokosmos Book 6, Bartok

First of all, the event space of figure 13 may be defined. (This is only one of many which could be used.)

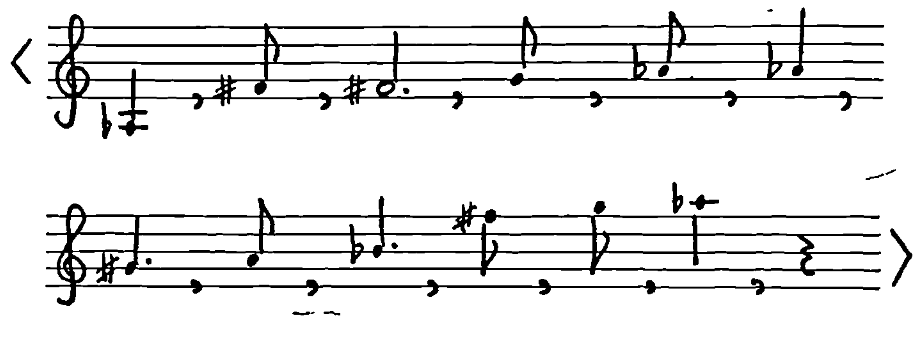
E =


figure 13

A pitch event space

|E| = 13

In the following chart, the number of times an event occurs is given in the first column, its relative percentage in the second, and corresponding probability in the third.

State	number of occurences	%	probability
1	1	3	·03
2	2	7	·07
3	1	3	·03
4	6	20	·2
5	4	13	·13
6	3	7	·07
7	1	3	·03
8	4	13	·13
9	3	10	·1
10	1	3	·03
11	1	3	·03
12	2	7	·07
13	2	7	·07
Total	30	100	1·00

The required probability assignment is in the final column:

P = (·03,·07,·03,·2,·13,·07,·03,·13,·1,·03,·03,·07,·07)

Some early experimenters with computer music thought that it was possible to reproduce melodic styles by analysing a given melody and using the results to generate new sequences. However, such a simple technique alone is quite inadequate, as was soon discovered. Nevertheless, the results of this type of test may be useful for study and comparison, and to generate material which has a restricted resemblance at the lowest level to the parent source.

The author has used these techniques to analyse intervallic content in recitatives from Passions by Handel and Bach.[3] The study produced some interesting observations which might otherwise have gone unnoticed. For example, it transpired that Handel, in general, used smaller intervals than Bach; and Bach's "Christus" recitatives contain many more intervals of the perfect fifth than might have been expected. Analysis by intervals is a particular use of difference distributions which are discussed in section 4.6 below.

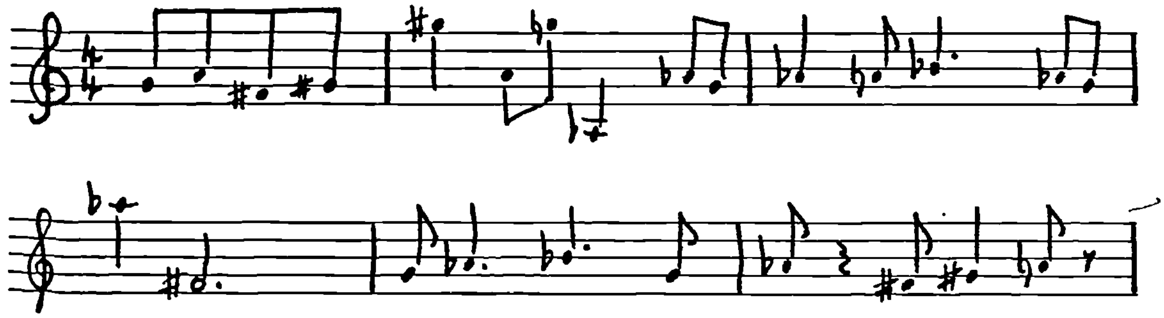
Probability assignments generated by analysis can be useful in composition, as long as the limitations are understood.

Example 4.4.1 continued

The event space and probability assignment of the example above, may generate a sequence such as the following:

4 8 2 5 12 8 11 1 5 4 6 8 9 5
4 12 3 4 7 9 4 5 13 2 6 8 13 ....

which transcribes into the form of figure 14 below.



A synthetic 'Chromatic Invention' based on the probabilities derived from Bartok's original in figure 12

figure 14

Any claim that this is representative of Bartok's style is clearly misleading. Such a small original sample is bound to generate a reasonably lively result with a sunerficial resemblace to Bartok. But as the generated sequence length is extended, the result lacks variety and structure, becoming formless and boring.

Defining an alternative event space, based on units two beats long for example, will produce a different set of similar characteristics.

The author has used the results of stochastic analysis of natural language as a compositional technique, but this was carried out using Markov Chain techniques where event probabilities vary according to their context, and so discussion of this is deferred until the next chapter.

4.5 Probability distribution defined by a function

One of the most interesting and productive uses of simple distributions involves using a particular function to define the probability assignment so forming a specific relationship between the individual probabilities. In this way coherent structures, with readily recognisable patterns may be generated.

Example 4.5.1

A regular distribution with a triangular shape may be generated using the function:

$$p(e_i) = \frac{2i}{n(n+1)} \qquad i = 1, \dots, n$$

where e_i refers to the i-th event

and n is the total number of events, as before.

Use of the above function will generate a simple arithmetic sequence of probabilities which all add up to unity.

So, if $n = 4$,

$$P = \left\{\frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}\right\}$$

If $n = 12$,

$$P = \left\{\frac{1}{78}, \frac{1}{39}, \frac{1}{26}, \frac{2}{39}, \frac{5}{78}, \frac{1}{13}, \frac{7}{78}, \frac{4}{39}, \frac{3}{26}, \frac{5}{39}, \frac{11}{78}, \frac{2}{13}\right\}$$

It can be quickly verified that in both of these cases,

$$\sum_{i=1}^{n} p(e_i) = 1$$

The use of such a distribution will generate a sequence with elements evenly biased towards events with a larger index, and event n occuring most frequently. With n equal to 4, in a sequence of 10 events, one would expect event 4 to occur four times, event 3 to occur three times, event 2 to occur twice and event 1 to occur only once. This bias shows clearly in a distribution diagram such as figure 15 which shows in graphic form the distribution expected if n = 12.

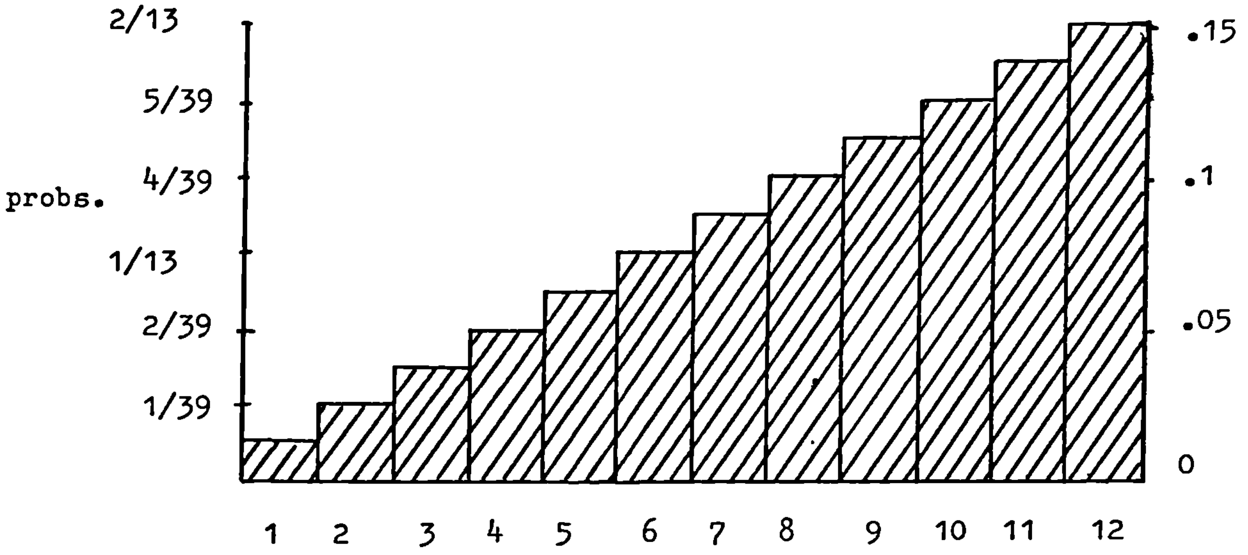


figure 15

A regular triangular-shaped distribution

If the probability assignment of order 12 as specified above is applied to the pitch event space consisting of the twelve notes of the chromatic scale above D, then a sequence such as that in figure 16 may be generated.

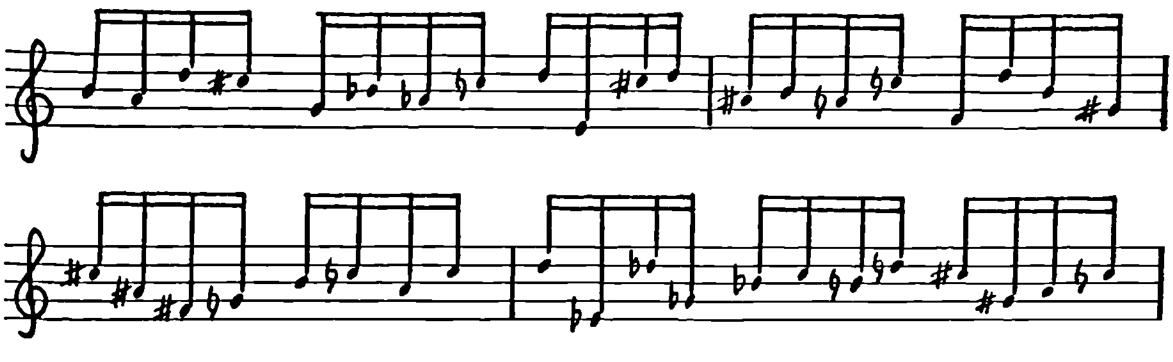


figure 16

A melodic sequence derived using a triangular-shaped distribution

The distribution used above is a regular triangular function. Such a regular distribution is unlikely to correspond to the event frequencies of any naturally occuring phenomena, but certain other standard distributions have been shown to model natural events closely. The sort of examples which statisticians cite to illustrate these distributions are: "The number of defective components in a batch", "The number of corpuscles in a given volume of blood", "Life expectancy at a given age" and even "The number of cavalry men killed by a horse-kick in a year" ! [4]

Distributions commonly used include the Binomial distribution and the Normal distribution, which are very similar in shape. An event probability graph of the Binomial distribution has the following form of figure 17.

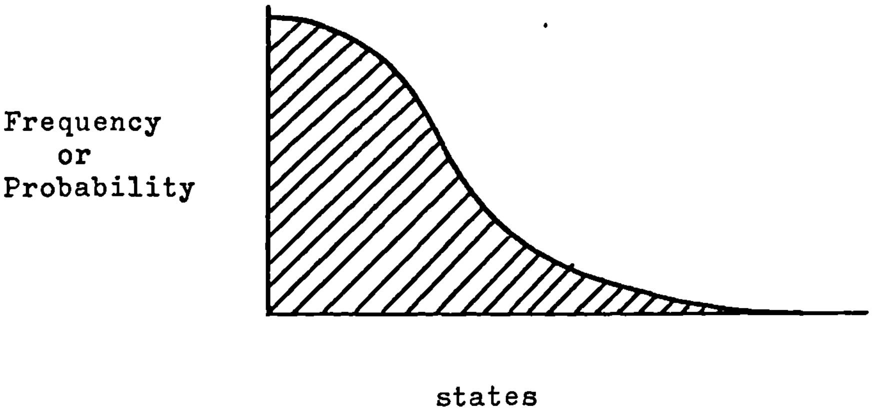


figure 17

The basic shape of the binomial distribution

Here there is a bulge of events all having close probabilities of occurance, which rapidly falls to a tail of unlikely or infrequent events.

It should be noted that as far as defining a generative probability assignment is concerned, turning the distribution around the other way (figure 18(a)) or having the bulge in the middle (figure 18(b)) amount to the same thing as using the original form of the distribution which may be reconstructed merely by re-naming and re-ordering the events.

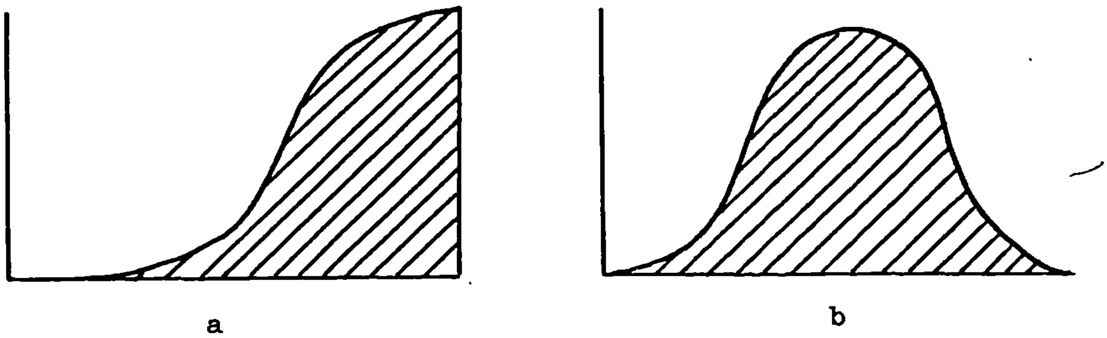


figure 18

Alternative forms of the binomial distribution

Another useful distribution is the *Poisson Distribution.*

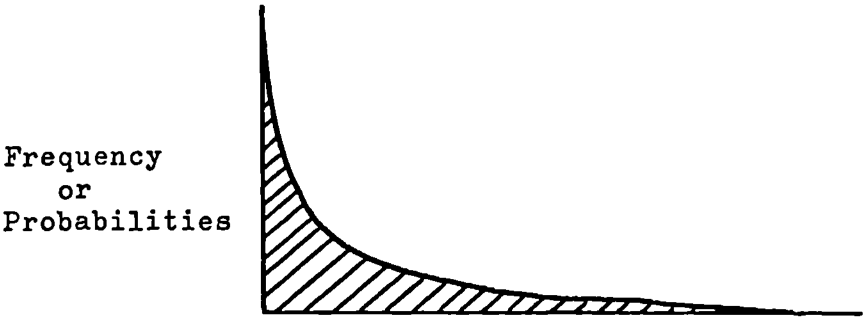


figure 19

The basic shape of the Poisson distribution

In this case, the probabilities immediately fall from the maximum and then gradually tail off as before. The function which may be used to derive the probabilities is:

$$p(e_i) = \frac{i^k}{i!} e^{-\lambda}$$

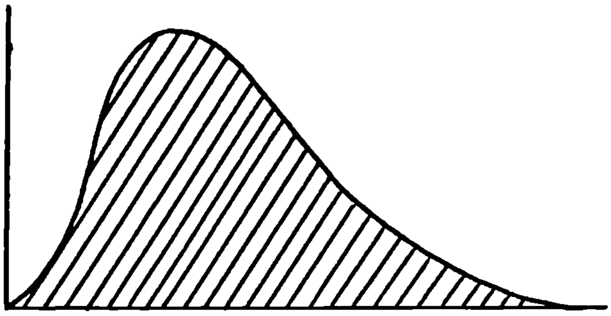
here, $i! = i\,(i-1)\,(i-2) \ldots 1$

$e^{-\lambda}$ is the exponential function, which models natural patterns of growth and decay

λ is a scaling factor which will determine the steepness of the curve and affect the overall mean density of events.

It has been shown that the Poisson distribution is ideal for modelling the occurence of isolated events in a continuum,[5] and it has consequently been applied successfully in musical composition. Xenakis has made use of the distribution in some of his works, notably *ACHORIPSIS* for orchestra,[6] and Barry Truax has used the Poisson distribution as the basis of his POD programs.[7] (POD is merely an acronym for Poisson Distribution.)

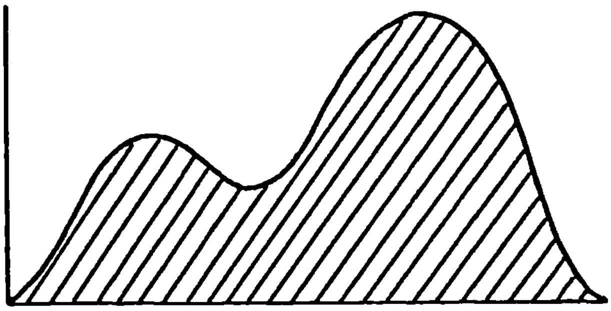
Other more complex functions include skew distributions where the rate at which probabilities approach or recede from the maximum may be different (figure 20).



A skew distribution

figure 20

There are also distributions with more than one peak (figure 21).



A two-peaked distribution

figure 21

However, the means of defining such functions can become rather complicated. It is simpler to divide the function into line segments, each of which is a triangular distribution. For small event spaces each probability may be specified individually as was done originally in section 4.3 .

When using a function for definition, it is not always necessary to define the separate probabilities of the assignment explicitly. The author has made use of the following function:

$$i \quad = \quad \mathrm{rnd}(\mathrm{rnd}(n))$$

which will produce an output with events distributed in a concave pattern which has a similar appearance to the Poisson distribution. This will be referred to as the *random decaying function*. Use of this function means that a required value can be generated directly, when required, and obviates the need to generate a complete probability assignment and laboriously work through it for each event to be generated (as described in section 4.3 above). This makes the computer operate much more efficiently, and is particularly useful if the number of events is large.

The asym om decaying function was used exclusively to generate material for the orchestral movement *FIRELAKE*. Its nature and application are discussed in the following chapter.

4.6 Probability distribution defined at random

It is possible when using a computer to employ random techniques to generate the probability assignment. This is done by generating a random number for each position in the assignment, (this will usually be a real number between 0 and 1 but it need not necessarily be so) and then calculating the sum of the numbers. The actual probabilities are then derived by working through the assignment and dividing each number by the sum. This ensures that the final probabilities total unity. So, if r_i is the random number generated in the i-th position of the assignment:

$$p(e_i) = \frac{r_i}{\sum_{i=1}^{n} r_i} .$$

Example 4.6.1

To generate an assignment of order 5.

Suppose the five random numbers are as below:

$$R = (\cdot 69 , \cdot 28 , \cdot 53 , \cdot 12 , \cdot 88)$$

Their sum is $2{\cdot}5$, so the resulting probability assignment becomes:

$$P = (\cdot 28 , \cdot 11 , \cdot 21 , \cdot 05 , \cdot 35)$$

and it can be seen that:

$$\sum_{i=1}^{5} p(e_i) = 1$$

It is possible to include in the assignment generating procedure a provision for forming a bias towards zero probabilities. Those events whose probability of occurence is zero are *null* events. Their function may seem to be redundant, but using this technique it is possible to define a number of different assignments on a fixed set of events. Thus, such a use of null events is equivalent to defining a set of different event spaces. Example 4.6.2 demonstrates this.

Even with such an apparently capricious composing method, with thoughtful definition of the event space, a reasonable degree of control may be achieved and interesting structures may be generated.

Example 4.6.2

An event space of order 12 for a short violin piece is defined in figure 22 (page 77). Two probability assignments are constructed at random:

$$P_1 = (\cdot 05 , \cdot 15 , 0 , 0 , \cdot 1 , \cdot 09 , \cdot 11 , 0 , \cdot 4 , 0 , 0)$$

$$P_2 = (0 , \cdot 1 , 0 , 0 , \cdot 2 , 0 , \cdot 2 , \cdot 1 , 0 , 0 , \cdot 3 , \cdot 1)$$

A distribution diagram of the same type as those used in the previous section helps to visualize the situation (figure 23).

P_1 is used to generate a sequence for the first section of the piece.

9 5 9 2 9 5 10 6 1 9 7 2 6 9 2 9 9 10 7 9

And then P_2 for a second section:

11 7 5 11 2 11 5 7 8 5 11 7 11 7 12 2 11 5 8 12

The result transcribes into figure 24 (page 79).

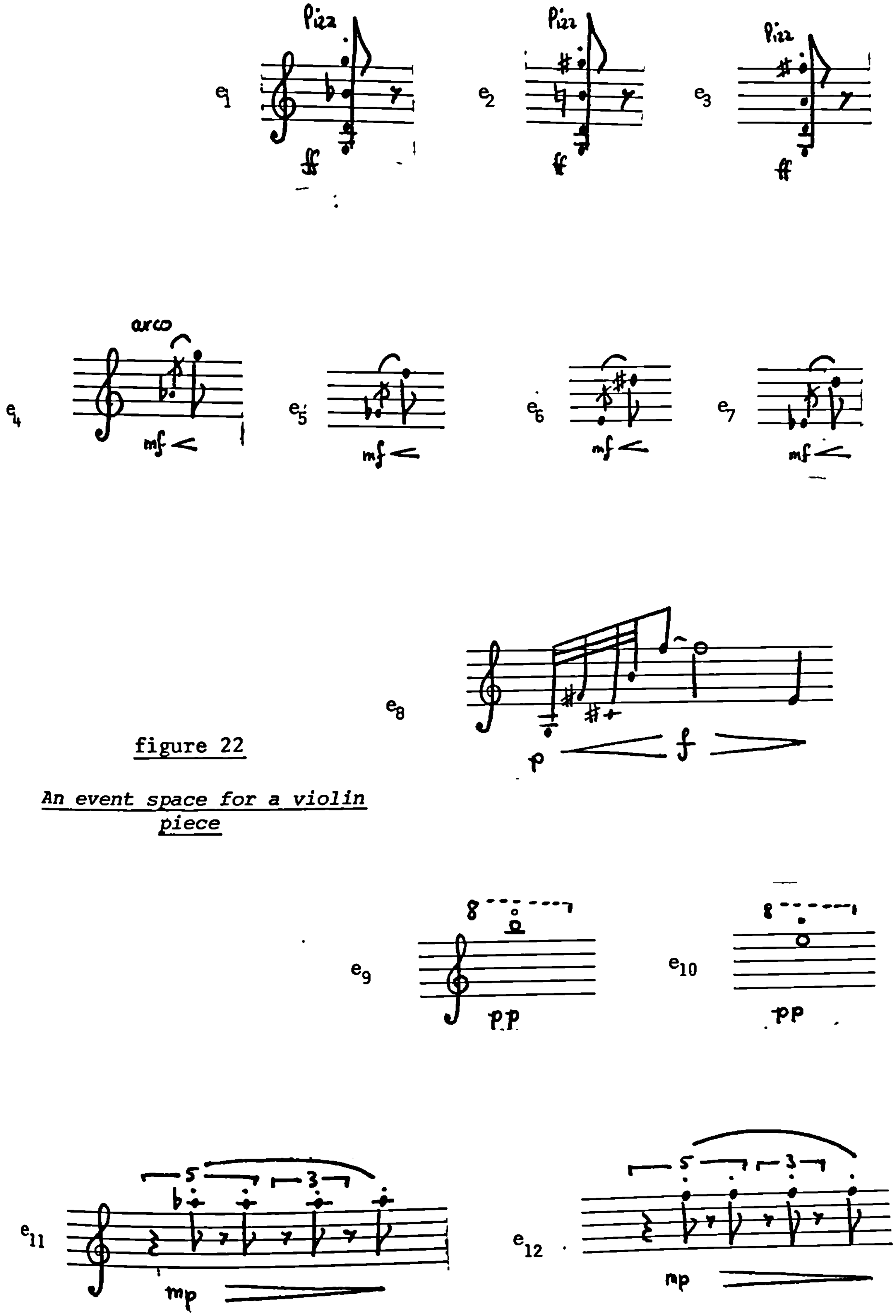


figure 22

An event space for a violin piece

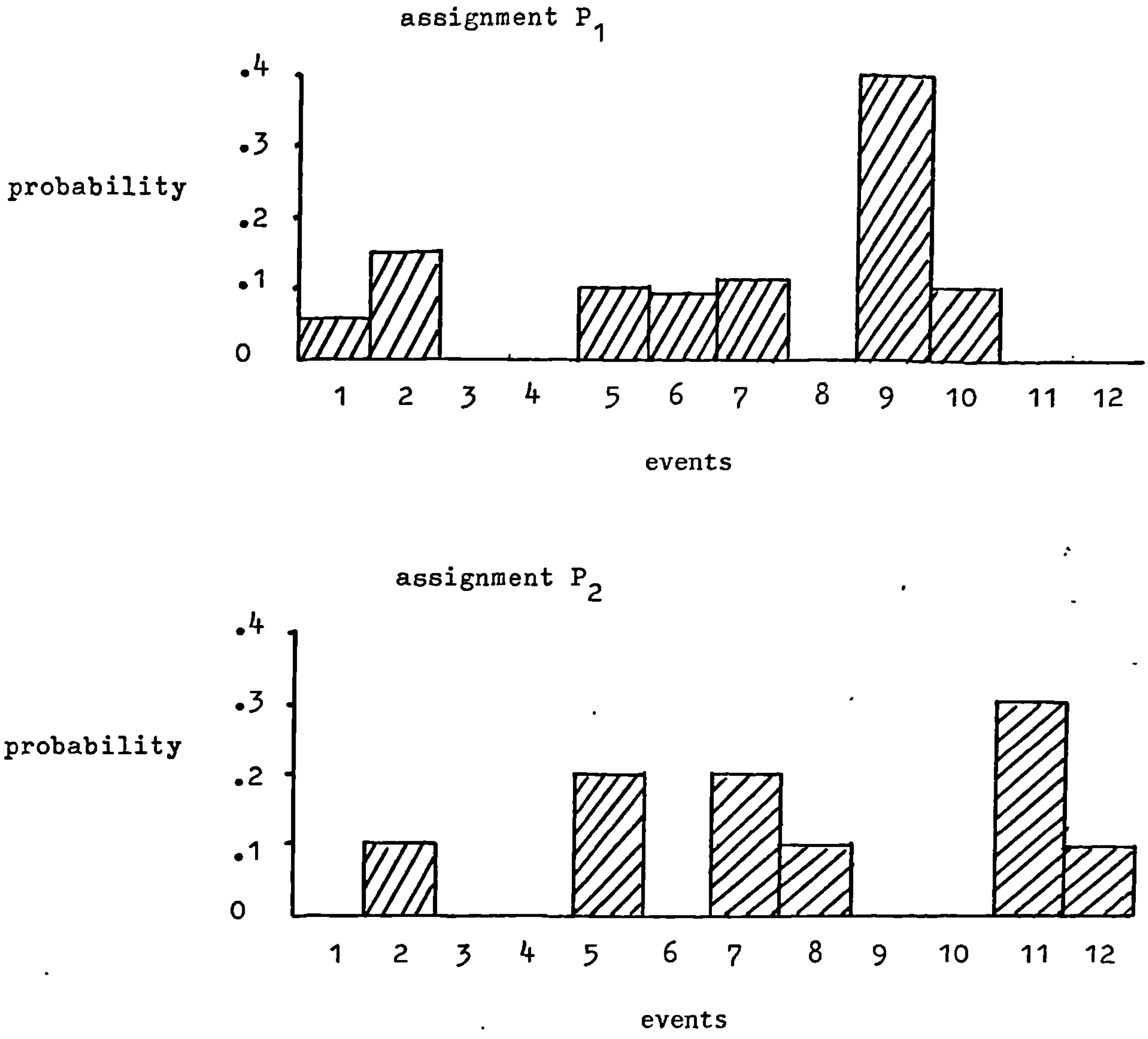


figure 23

Two probability assignments for a violin piece

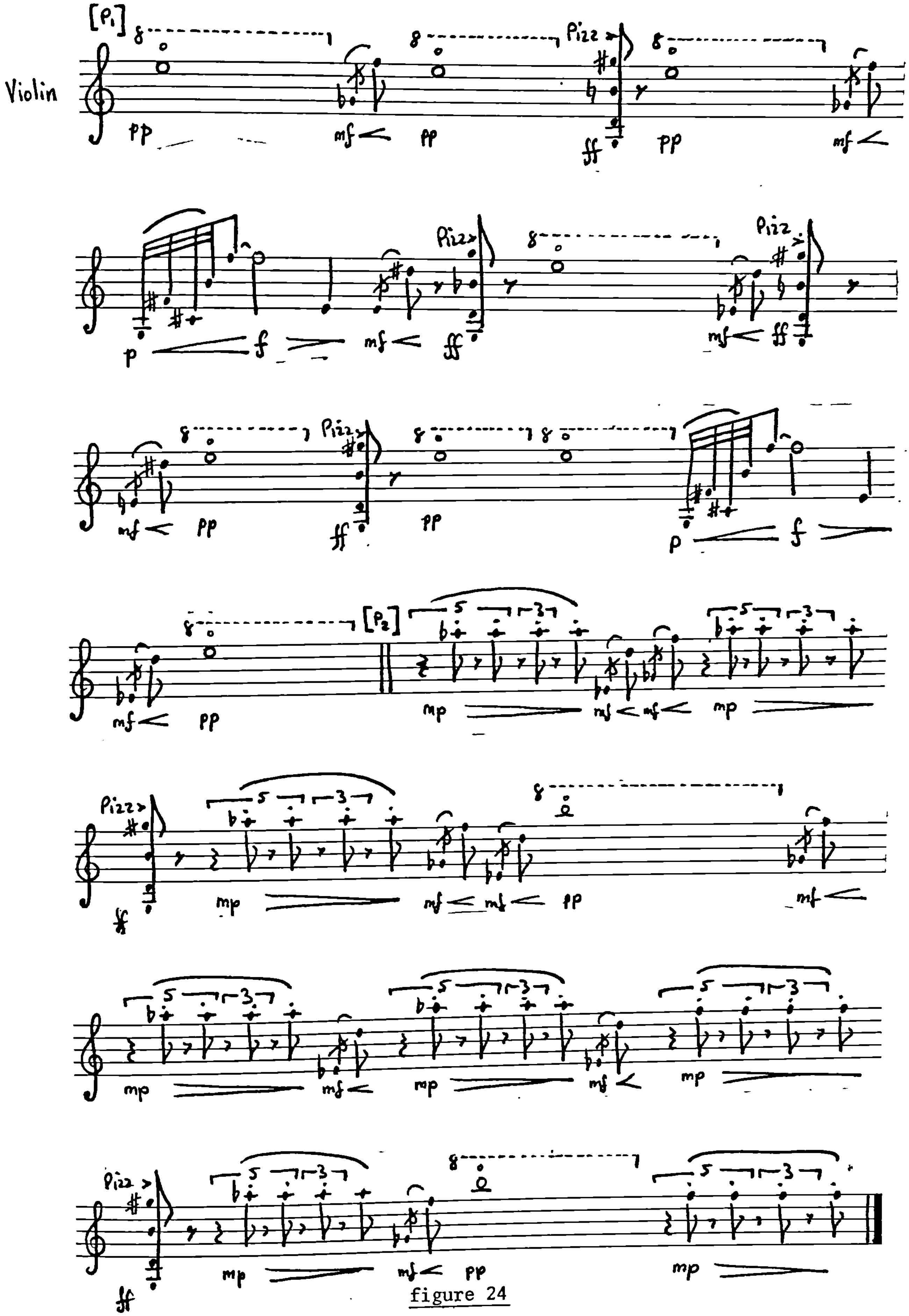


figure 24

Short violin piece formed by the application of the probability assignments of figure 23 over the event space of figure 22

In earlier work the author has made much use of probability assignments formed at random,[8] but these have mostly been in the context of more complex Markov systems which are to be discussed in chapter 6.

4.7 Difference Distributions

The Markov Chain systems to be considered shortly take into account the past history leading up to the occurence of an event; its occurence probability depends on the immediately preceding event. *Difference distributions* form a bridge between the simple distributions which have been considered and Markov chains by considering the frequency or probability distribution of *differences* or *intervals* between event indices. These can be particularly useful in analysis, but present difficulties when used as a generative model.

Example 4.7.1

Consider the following sequence of events, from an event space of order 6.

$$6 \quad 5 \quad 1 \quad 1 \quad 3 \quad 4 \quad 6 \quad 5 \quad 1 \quad 2 \quad 1$$

It is possible to construct another sequence which specifies the *difference* between each event.

$$-1 \; -4 \; \; 0 \; +2 \; +1 \; +2 \; -1 \; -4 \; +1 \; -1$$

A *difference space* D_E can then be constructed.

$$D_E \quad = \quad (-4 , -3 , -2 , -1 , 0 , +1 , +2)$$

The probability distribution over the difference space, P_{D_E} , is defined by counting the number of occurences of each difference, and dividing by $|D_E|$, the total number of differences. So,

$$P_{D_E} = (\cdot 2 , 0 , 0 , \cdot 3 , \cdot 1 , \cdot 2 , \cdot 2)$$

In this way a consideration of the "up and down" movement of a sequence may be incorporated into the model, and so some formulation of a state's contextual function is maintained.

The best method of representing types of difference distributions is in the context of Markov Chain systems and so further discussion is reserved until later.

CHAPTER FIVE

APPLICATION IN COMPOSITION (I) - FIRELAKE

5.1 Composing semantics

FIRELAKE is the central section of a larger orchestral work: *WINDOWS INTO ETERNITY* which portrays various visions in the 'Apocalypse' or 'Revelation' of St. John:

> "... the devil was thrown into the lake of fire and brimstone where the beast and the false prophet were, and they will be tormented day and night for ever." [1]

The basic musical plan was to use the strings as a background, maintaining a continuous shuffling profile to represent the bubbling red surface of the lake, from which yellow and orange splashes of woodwind colours came flickering out in tongues of flame; all this to be periodically punctuated with white tormented shrieks from the brass and deep aeonic rumbles from the timpani.

This basic conceptual design is represented in figure 25.

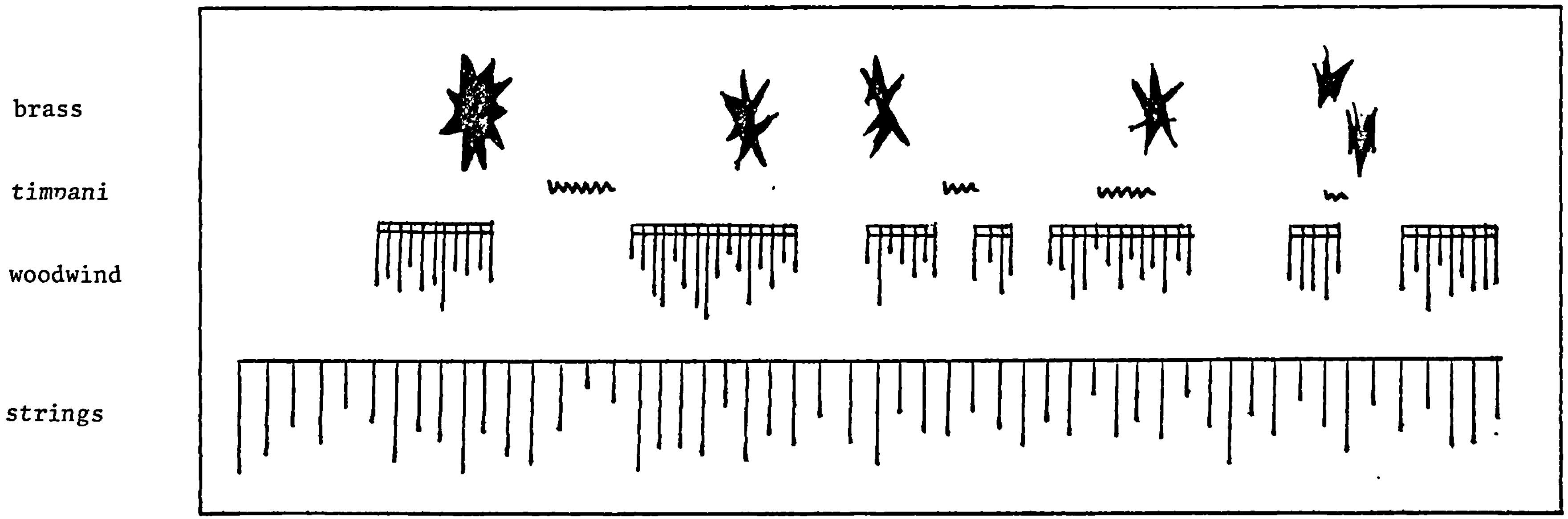


figure 25

Basic conceptual structure of FIRELAKE

5.2 The Random Decaying Function

It was decided to make use of the Random Decaying Function (abbreviated to RD-function) which was introduced above (section 4.5). The particular form of the function used was the following:

$$i \quad = \quad \mathrm{rnd}(\mathrm{rnd}(n) + 1)$$

This will deliver a random integer in the range of 0 to n-1 inclusive, and can be used to index a total of n events. The indices will be biased towards 0 . Figure 26 shows the distribution formed by this function over an event space of order 10.

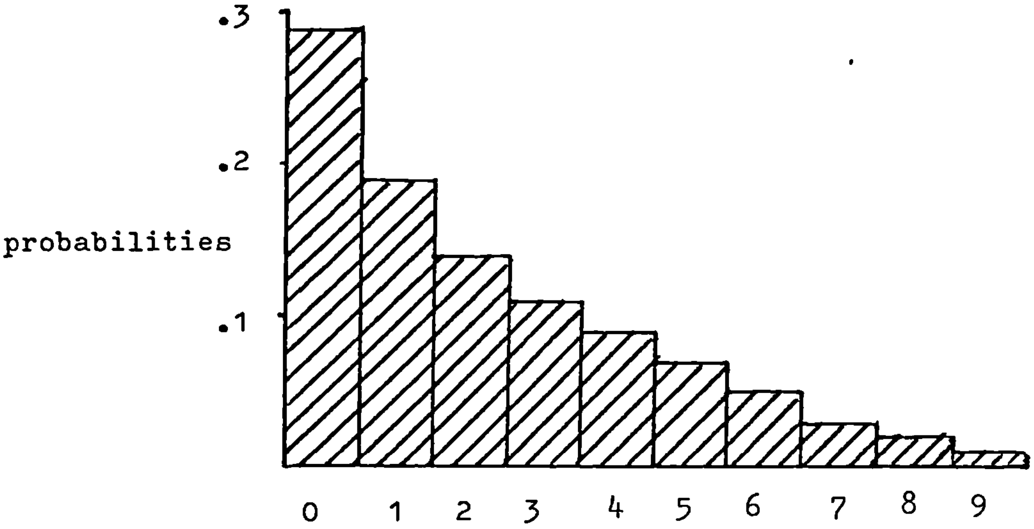


figure 26

The random decaying function

The effect of the use of this function is to form a background of frequently occuring events, with a bias towards a particular event, into which the less frequent events only occasionally intrude. This maintains an optimal balance between repetition of familiar material and the introduction of unexpected, intermittant extremes.

The nature of the practical application of the function was varied to suit the musical context and the effect being sought. It was used to control all of the main compositional parameters, including pitch, duration and intensity in a number of different ways. These, and the different event spaces over which the functional assignment was applied, are detailed for each instrumental class in the sections which follow.

The overall jurisdiction of the function provides for a high level of structural unity and integration.

5.3 Application of the distribution

STRINGS

The strings play in quavers throughout, in unison or octaves. 'G' was chosen as the tonal centre, on which the pitch structure is built, and so, using the RD-function to derive the pitches means that the probability of a particular pitch occuring reduces the higher above 'G' it is. Figure 27 below is an example of the type of pitch sequence this may produce.

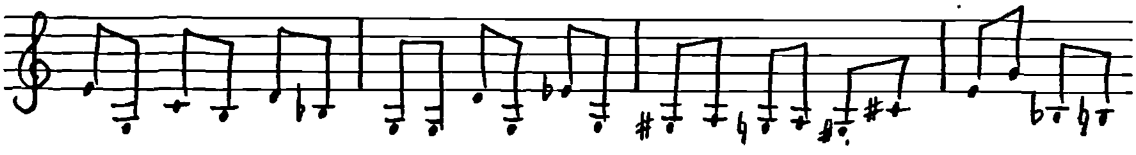


figure 27

A pitch sequence produced by the RD function

Even though the individual notes are of constant length, the RD-function was used to derive sectional durations. It was decided that the overall density of the string writing should be 0·75, that is each instrumental part should on average rest for approximately one quarter of the time. Three calls were made to the RD-function and the sum used to determine the length of a sequence of pitches in bars, and then one call to determine the length of following silence. This means that the silent sections tend to be short, with only occasional longer rests.

Reference should be made to the full score in Appendix A.

The RD-function was also used to define intensity, on a six point scale from ff to pp , with the bias towards ff , and the length of time for which each intensity state was to apply, in terms of number of whole bars and number of additional beats.

It should be pointed out that at least as far as the strings were concerned "zero' lengths, which are in fact the most common, will not appear at all and so the actual distribution in this case is a little different from the RD-function which otherwise governs the compositional structure.

WOODWIND

The woodwind also play in unison or octaves, but in semiquavers and in intermittent groups of notes. The density of sound in relation to the whole playing time was chosen as 0·5 , so the RD-function was called to determine the length of pause, in quavers, and then again to determine the length of note groups, also measured in quavers. 0's were ignored. Figure 28 below shows the sort of rhythmic structure which is produced.

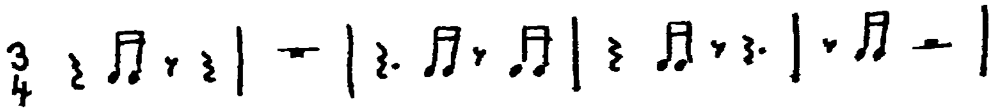


figure 28

A rhythmic sequence produced by the RD function

This resulted in twenty five note groups being formed, separated by rests. In addition, a short delay was introduced before the first woodwind entry.

The pitches were chosen in the same way as for the strings. The instrumentation for each of the twenty five note groups was determined by working through each instrumental part, and using the RD-function to determine for how many groups the instrument played, and for how many groups it rested. Zero's were included this time, so where an instrument plays for a zero number of note grouns, it ends up enjoying long rest periods. The resulting woodwind instrumentation scheme is represented in figure 29 (page 90).

BRASS

The RD-function determined the pattern of entries, with a density of 0·25 , i.e. the brass plays for a total of only one quarter of the time. The function was also used to determine the rhythm pattern for each note group. 0 indicated an acciaccatura, and the values 1 to 6 represented note lengths in semiquavers. The following is an examnle of the rhythm of the note groups so produced:

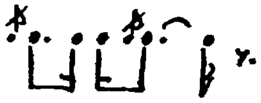


figure 30

A rhythmic fragment produced with a differeht method of applying the RD function

Again, a delay was introduced before the first brass entry.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
flute 1	✓	✓							✓																
flute 2 (piccolo)			✓	✓	✓	✓	✓	✓												✓			✓		
oboe 1													✓								✓	✓	✓	✓	✓
oboe 2	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
clarinet 1	✓	✓	✓	✓					✓	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
clarinet 2	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	
bassoon 1								✓	✓			✓	✓	✓		✓	✓	✓	✓	✓			✓	✓	✓
bassoon 2	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓										✓			

figure 29

Woodwind instrumentation scheme for FIRELAKE

The function was then used to determine the number of different pitches required in each chord represented by the notes in the rhythmic pattern, the values of the pitches in the chord, and the instruments available to play them. The actual allocation of pitches to instruments, and choice of register, was determined by hand so as to achieve the most easily playable arrangement for each instrument.

The simple constraints could easily have been programmed for the computer to do the job, but it would have taken longer. The computer should always be used as a servant, to aid composition; not as a master, used for its own sake.

The brass play fortissimo throughout.

TIMPANI

The timpani entries were determined at a density of 0·125, with rhythmic patterns constructed in the same way as for the brass. The tonal centre 'G' is used as the only pitch, and the player is instructed to play fortissimo.

5.4 Conclusions

The full score of *FIRELAKE* can be found in Appendix A.

Having heard the work performed, the author is generally quite satisfied with the result. Use of the RD-function produces a structural balance and integrity which is quite easy to perceive. In this application the composer had complete control of the structural development of the piece, and was able to assess the effect of various alternatives before deciding upon the most effective interpretations and applications of the function being used, which was acting very much as servant rather than master of the operation; although having decided to use the function, there is a sense in which decisions are made in order to make it work effectively. Its nature and presence can therefore determine and mould the direction of compositional activity. But these are none the less very stimulating constraints within which to work.

When the work was introduced to the orchestra giving the first performance, they were told that one section of the seven in the complete piece was written with computer assistance. After playing the work through they were invited to suggest which section it was. The composer thought that it was obvious, but surprisingly no one identified the section correctly! The reaction judged on such an informal basis of course has little value as a valid statistical test. The players' expectations of what computer music ought to sound like could vary dramatically. But the reaction does seem to demonstrate that the section blends into the work as a whole, and is not a compositional cuckoo.

FIRELAKE was first performed at Strasswalchen near Salzburg on 31st July 1979.

CHAPTER SIX

STOCHASTIC PATTERN GENERATION (II)

MARKOV CHAINS

The patterning systems described in chapter four were all derived from simple probability distributions, which remain constant over a period of time. Much more sophisticated control mechanisms may be constructed, however, by taking into account the context of an event in a sequence, and making the probability of the occurence depend upon the event which preceeded it. The analysis and generation of such sequences were first considered by the Russian mathematician A. A. Markov (1856 - 1922) who was engaged in a study of the alternation of vowels and consonants in Pushkin's "Eugene Onegin".[1] This is an interesting example of how essentially artistic or literary directed ends led to the development of a sophisticated mathematical tool which has had a profound influence in every aspect of mathematical statistics, electronic engineering and even economics.

6.1 Matrix of transition probabilities

If i and j are the indices of events from an initial event space, and E_m is the event index of the m-th event in the generated sequence, then the following relation formally expresses the dependence of each event's probability on its antecedent:

$$p_{ij} \quad = \quad p(E_{m+1} = j \mid E_m = i)$$

In other words, p_{ij} represents the probability of event i being followed by event j .

It is usual to express the probabilities p_{ij} in the form of a matrix. The rows represent the 'current events', for each of which there is a separate probability assignment to determine the next event. It follows that the matrix will be a square matrix. It will have the following form:

$$P \quad = \quad \begin{pmatrix} p_{11} & p_{12} & \cdot & \cdot & \cdot & p_{1j} & \cdot & \cdot & \cdot & p_{1n} \\ p_{21} & p_{22} & \cdot & \cdot & \cdot & p_{2j} & \cdot & \cdot & \cdot & p_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ p_{i1} & p_{i2} & \cdot & \cdot & \cdot & p_{ij} & \cdot & \cdot & \cdot & p_{in} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ p_{n1} & p_{n2} & \cdot & \cdot & \cdot & p_{nj} & \cdot & \cdot & \cdot & p_{nn} \end{pmatrix}$$

It is clear that the sum of each row must be 1 :

$$\sum_{j=1}^{n} p_{ij} = 1 \quad i = 1 , 2 , \ldots , n$$

And the sum of all the probabilities must therefore be equal to the order of the event space:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} = n$$

Such a matrix as this is called a *matrix of transition probabilities* or a *stochastic matrix.*

Example 6.1.1

The following stochastic matrix defines conditional probabilities on three events:

$$P_1 = \begin{bmatrix} \cdot 8 & 0 & \cdot 2 \\ 0 & 0 & 1 \\ \cdot 2 & \cdot 8 & 0 \end{bmatrix}$$

This matrix defines a situation where event 1 will tend to be followed by itself and occasionally by the third event, event 2 will always be followed by event 3 , event 3 will mostly be followed by event 2 , but occasionally by event 1 . The following is an example of the sort of sequence this matrix will produce:

$$e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_3\ e_2\ e_3\ e_2\ e_3\ e_2\ e_3\ e_2\ e_3\ e_2$$
$$e_3\ e_2\ e_3\ e_1\ e_1\ e_1\ e_1\ e_1\ e_3\ e_2\ e_3\ e_2\ e_3\ e_2\ e_3\ e_1\ e_1\ e_1\ e_1$$

Patterns generated by this matrix will always have long sequences of one repeated event followed by sequences of two alternating events. This may be interpreted in musical terms by using the three-event space defined in figure 31(a) producing the sequence transcribed in figure 31(b).

Example 6.1.2

Changing the probabilities.

Using the same event space, a different stochastic matrix may be defined:

$$P_2 = \begin{bmatrix} 0 & \cdot5 & \cdot5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

In this case, both event 2 and event 3 will always be followed by event 1 , but event 1 may be followed by either event 2 or event 3 with equal probability. A sequence such as the following may be produced:

$$e_1\ e_2\ e_1\ e_3\ e_1\ e_2\ e_1\ e_2\ e_1\ e_3\ e_1\ e_3\ e_1\ e_3\ e_1\ e_2\ e_1\ e_3\ e_1\ e_2\ e_1$$

In other words, events 2 and 3 alternate with event 1 . Transcribed into musical terms, using the same event space as example 6.1.1 above, this sequence will produce figure 32. Even though this example is constructed from the same simple material as figure 31(b), it can be seen that the musical effect is quite different.

It is often useful in helping to visualise the structure of a matrix, to use an event-relation diagram of the type associated with the study

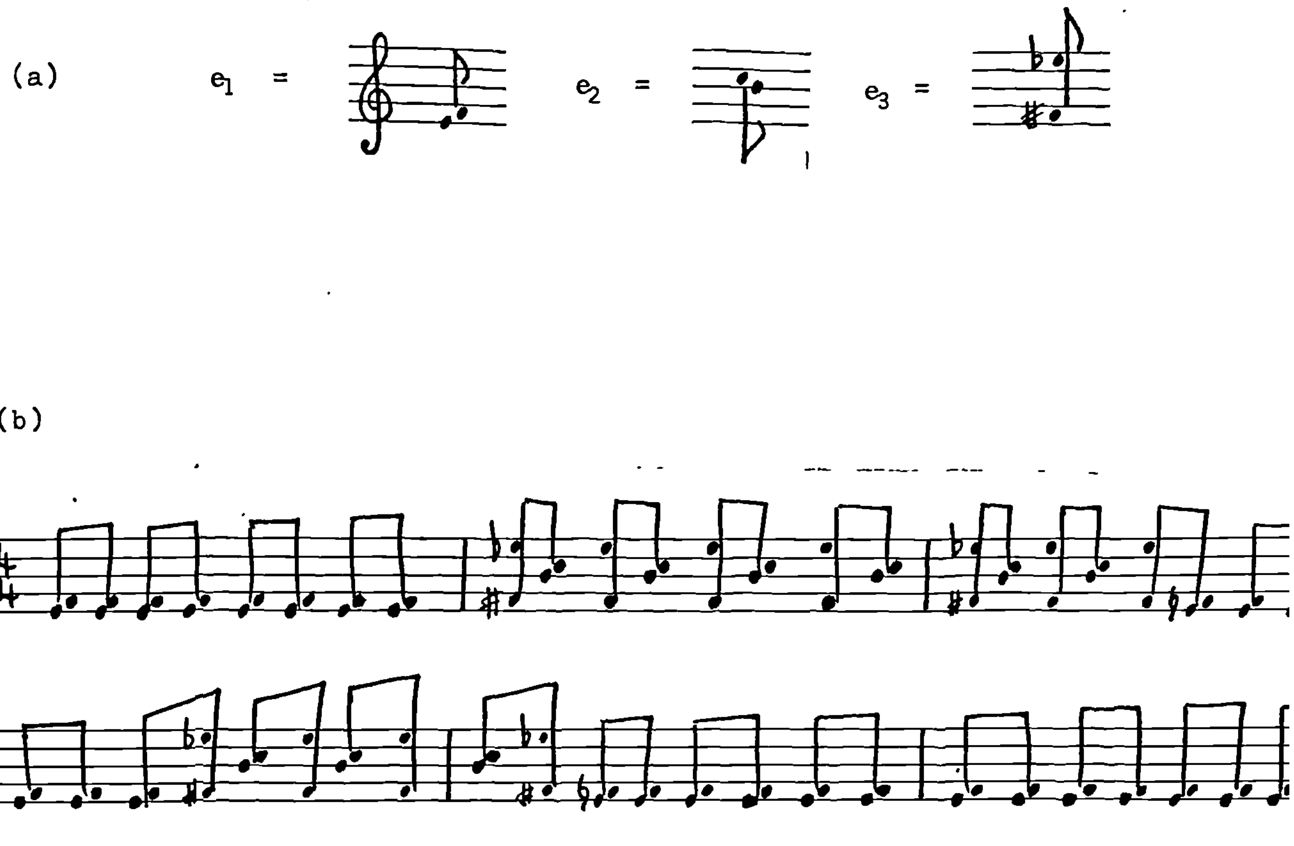


figure 31

An event space and event sequence formed by the application of a Markov Chain

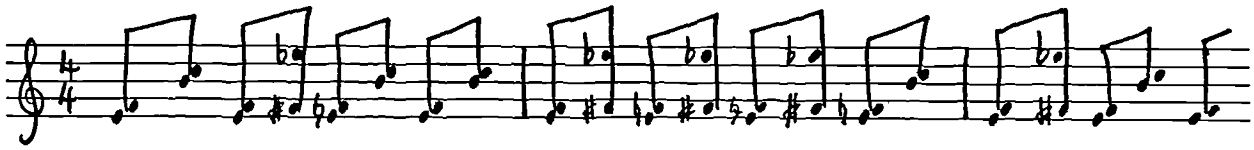
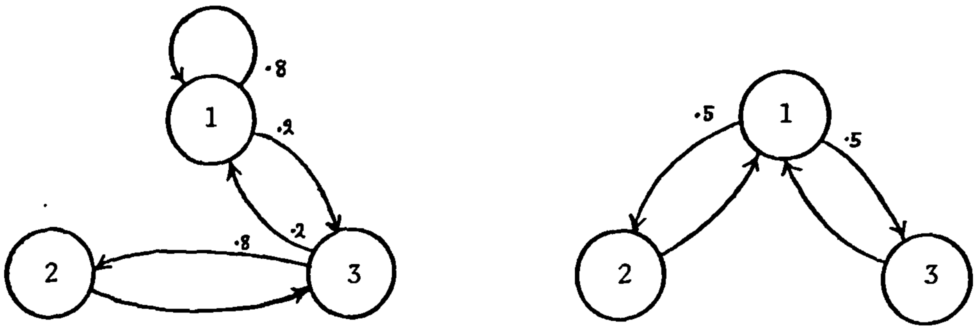


figure 32

An event sequence formed by applying a different Markov Chain to the same event sequence as figure 31 of finite automata, with lines representing the possibilities of changing from one event to another, as defined in the matrix.

Example 6.1.3

The matrices in examples 6.1.1 and 6.1.2 will apnear as in figure 33, which gives a clear picture of the structures involved.



Example 6.1.1

Example 6.1.2

figure 33

Event relation diagram of two simple Markov Chains

The method of using a matrix to generate a sequence is an extension of the procedure described in section 4.3. The initial event may be chosen at random or specified by the user; the row corresponding to this event is then identified in the matrix, the probability assignment specified in that row is used to derive the next event, event j, using the technique described in section 4.3, then event j becomes the current event, the j-th row is identified, and the process is repeated all over again many times to produce the required length for the sequence.

In a similar way to the process described in section 4.4, it is possible to analyse given sequences to produce material for use in composition.

As before, this is best explained with an example. The techniques used in composing the choral work *TEXT YEARS* will be described.

Example 6.1.4

TEXT YEARS : Markov Chain analysis in composition.

TEXT YEARS is a choral work which explores the massed effect of a large number of voices reading special texts which were carefully prepared to generate distinctive textures. The sound of letters in a sequence of synthetic text is strongly dependent on an individual letter's context. This is particularly true of vowel combinations. For example, the sound of the letter "o" is different each time it appears in the following invented words, when spoken by someone who is used to standard English pronunciation:

moso   moosoop   kolokos   moalo

It therefore seemed appropriate to make use of Markov Chain techniques to form the texts to be used in the composition since simple probabilities would be inadequate in providing consistent sonic textures.

Short contrasting fragments of text were constructed, being designed to produce particular effects. For example, "dum dum da duma da duma dum ..." will sound like plucked strings, "konng gonngong ..." like bells and "krak tak a kraka ..." like spitting, crackling fire. Each fragment of text was analysed to construct a stochastic matrix which was then used to generate a substantial amount of textural material which would be consistent in its probability structure, and hence in its overall stochastic sound texture. The raw material produced in this way was drawn upon to arrange and develop contrasting textural blocks split amongst four groups of performers in the final score.

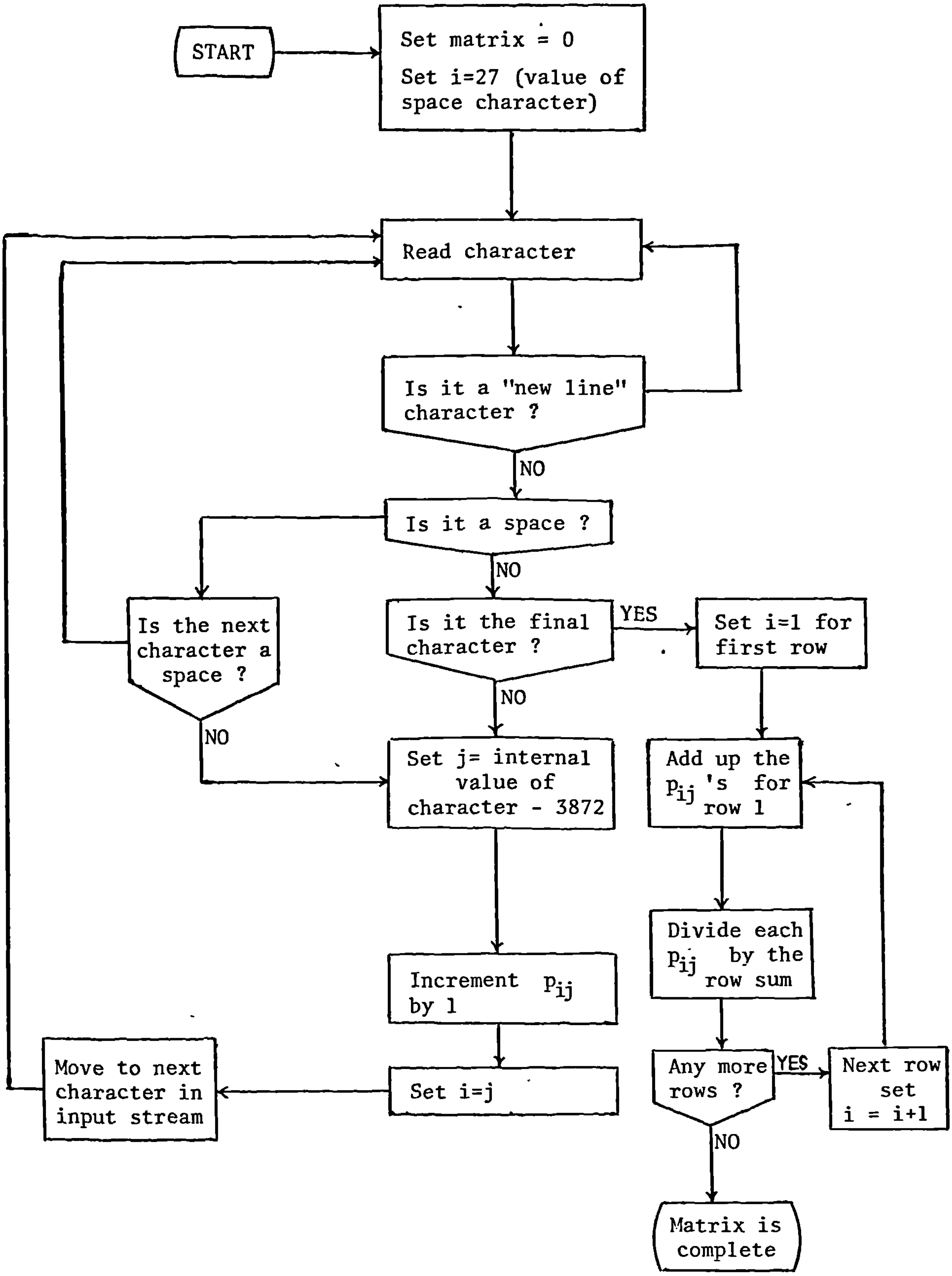
The program for analysing and synthesizing texts using the Markov chain model may be found in Appendix C. This is written in ALGOL 60. A flow chart showing the operation of the first part of the program, which analyses the input text and generates a stochastic matrix from it, is given in figure 34. The second half of the program, which uses the matrix to generate sequences, operates in the way already described above (page 99).

The complete score of *TEXT YEARS* is in Appendix B.

TEXT YEARS was first perfomed in the New Hall of the City University London on 24th April 1978, and subsequently at the ICA, London on 17th December 1978.

The program used for *TEXT YEARS* was also used to construct the text for the composition *DUTCH COURAGE*. This program is not limited to text analysis and synthesis, but may also be used on any sequence of symbols, for example the notes in a melody, providing the input is encoded into alphabetic form.

Markov chains have been the subject of a considerable amount of mathematical study.[2] Different types of events, matrices and patterns have been classified and studied, and a number of useful results have


START
Set matrix = 0
Set i=27 (value of space character)
Read character
Is it a "new line" character ?
NO
Is it a space ?
NO
Is the next character a space ?
NO
Is it the final character ?
YES
Set i=1 for first row
NO
Set j= internal value of character - 3872
Add up the p_{ij} 's for row 1
Divide each p_{ij} by the row sum
Increment p_{ij} by 1
Move to next character in input stream
Set i=j
Any more rows ?
YES
Next row set i = i+1
NO
Matrix is complete

Flow chart explaining the basic operation of the TEXT YEARS program

figure 34

been obtained. Some of the basic results are summarized in the following sections, and their musical implications are considered.

6.2 Classifying events

An awareness of the different types of events, in terms of their function or status in a Markov Chain, can help in understanding the structure of a chain formed by analysis or random techniques. This in turn can help in the definition of chains, either for specific tasks, or by making use of a limited degree of semi-random control, but at the same time aiming to produce a coherent result. A theory built up from event analysis can help to describe and predict certain properties of the resulting event sequences.

6.2.1 Accessible and communicating events

An event j is said to be *accessible* from event i if j can be reached from i in a finite number of steps. If j is accessible from i , this can be written:

$$i \longrightarrow j$$

If j can be reached from i in one step, it is said to be *directly accessible.*

If j is accessible from i , and k is accessible from j , then it can be seen that k is accessible from i , i.e.

$$i \longrightarrow j \quad \text{and} \quad j \longrightarrow k \quad \Longrightarrow \quad i \longrightarrow k$$

This is the *transitive* property of the accessibility relation.

Example 6.2.1.1

Referring to the matrix P_1 of example 6.1.1 (page 96), it can be seen that event 3 is accessible from event 1, and event 2 is accessible from event 3, and so event 2 is accessible from event 1 (but not directly accessible). This can be seen quite clearly in the event diagram of figure 33 (page 99). Notice that in this case all the events are accessible from themselves, i.e.

$$i \longrightarrow i \qquad\qquad i = 1 , 2 , 3$$

But only event 1 is directly accessible from itself.

It should be pointed out that if event j is accessible from event i it does not follow that event i is accessible from event j, so the accessibility relation does not have the *symmetric* property.

Example 6.2.1.2

Consider the Markov Chain defined by the following stochastic matrix

$$P_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & \cdot 2 & \cdot 8 \end{bmatrix}$$

The event diagram of figure 35 helps to visualize the structure:

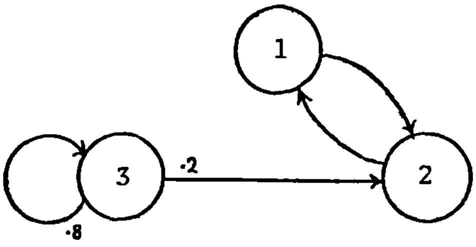


figure 35

Event relation diagram of a simple Markov Chain with accessible and inaccessible events

Both events 1 and 2 are accessible from event 3 , but event 3 is not accessible from either of events 1 or 2 . Events 1 and 2 are both accessible from each other.

It can be seen that a sequence produced by this chain, if begun with event 3 , will continue with a string of 3's , to be eventually followed by events 1 and 2 alternating.

3 3 3 3 3 3 3 2 1 2 1 2 ....

Once event 2 has occured, event 3 cannot occur again.

If the two events i and j are both accessible from each other, then they are said to *communicate*. In examples 6.1.1 and 6.1.2 (figure 33, page 99) all the events communicate with each other. In example 6.2.1.2 events 1 and 2 communicate, event 3 communicates only with itself.

It can be seen that the communication relation is both transitive and symmetric.

Any event which communicates with another event, must communicate with itself. This is known as the *reflexive* property and it follows immediately from the symmetric and transitive properties.

$$\text{If } i \longrightarrow j \text{ and } j \longrightarrow i$$
$$\text{then } i \longrightarrow i \text{ and } j \longrightarrow j$$

The communication relation is represented by a double headed arrow:

$$i \longleftrightarrow j$$

The three properties of the communication relation which have been established can thus be summarized as follows:

1. Reflexivity: $i \longleftrightarrow i$
2. Symmetry: If $i \longleftrightarrow j$ then $j \longleftrightarrow i$
3. Transitivity: If $i \longleftrightarrow j$ and $j \longleftrightarrow k$ then $i \longleftrightarrow k$

Any relation which exhibits these three properties of relexivity, symmetry and transitivity is known as an *equivalence relation.* The fact that the communication relation is an equivalence relation is very significant and will be developed in the following section (6.3).

6.2.2 Recurrent and transient events

If, and only if, after an event i has occured, a subsequent return to this event is certain, then event i is said to be *recurrent.* If, on the other hand, starting from event i, there is a chance that the process will not return to that event, then it is said to be *transient.* Thus all events may be classified as recurrent or transient. It can be seen that a Markov chain cannot consist entirely of transient events, but it is possible that it may contain only recurrent events.

In the Markov Chains of examples 6.1.1 and 6.1.2 (figure 33), the process may move freely between each event. The fact that an event will eventually re-occur, if the process continues for long enough, is certain. Thus all of the events in both examples are recurrent.

In the Markov Chain of example 6.2.1.2 (figure 35), once event 3 has occured, it may occur again, but if the process should move to event 2, event 3 cannot occur again. Thus event 3 is a transient event. Events 1 and 2 will continue to occur however, and thus are recurrent events.

An important result follows from these definitions. *If two events i and j communicate, and i is recurrent, then event j is also recurrent.* This should be fairly easy to see. The following argument develops a brief intuitive proof.

Given that events i and j communicate and event i is recurrent, let a process begin with event i and then proceed to event j. Suppose that after continuing from this point a return to event j is impossible, then a return to event i must also be impossible since event j can always be approached via event i. But this contradicts the initial condition that i was recurrent, hence a return to j must be possible and so event j is also recurrent.

Similarly, as a corollary to the above result, it can also be shown that *if two events i and j communicate and event i is transient then event j must also be transient.* For, if event i is transient and the process moves to such a position that it cannot return to event i, then event j must also be inaccessible otherwise the process could return to event i via event j with which it communicates. Hence event j is also transient.

A more rigorous mathematical proof of the above results may be found in the literature.

Again, the significance of these results is made explicit in section 6.3.

6.2.3 Periodic and aperiodic events

If starting from an event e_i in a Markov Chain, t is the number of steps taken to return to e_i, and a return to e_i can only occur at multiples of t, and t is the largest integer with this property, then t is called the *period* of event e_i. If t is greater than 1, then event e_i is said to be *periodic*. If t is equal to 1, then event e_i is said to be *aperiodic*.

Referring to example 6.1.1 again (page 96); event e_2 can recur after two or four steps, and also after any number of steps above that depending upon the number of times the process reneats event e_1. Similarly, event e_3 can recur after 2, 3, 4, ... steps, and event e_1 can recur after one or more steps. Thus, all of these events are aperiodic.

Referring to example 6.1.2, event e_1 will recur after two steps, events e_2 and e_3 will recur after 2, 4, 6, ... steps. Thus each event in this Markov Chain has a period of length 2. Examination of the typical sequence generated in this same example clearly shows the periodic effect.

In example 6.2.1.2, event e_3 will recur only after one step and is thus aperiodic. Events e_1 and e_2 always recur after two steps and thus each has a period of length 2.

Another important result follows from the definition of periodicity. *If two events e_i and e_j communicate, then e_i and e_j have the same* *period.* The proof of this result is quite complicated and beyond the scope of this thesis, but the examples given above demonstrate that it is indeed the case.

It is particularly useful to notice that if event e_i is directly accessible from itself, i.e. if $p_{ii} > 0$, then any event with which it communicates will be aperiodic. This is the case with event e_3 of example 6.1.1, but not with event e_3 of example 6.2.1.2 which does not communicate with any other event.
An event which is both recurrent and aperiodic is known as an *ergodic* event.

6.2.4 Absorbing events

An event e_i is said to be an *absorbing* event if and only if $p_{ii} = 1$. In terms of the sequence generated by a Markov Chain containing the absorbing event e_i, this means that once event e_i has been reached it will continue to be repeated and no other event can subsequently occur. Clearly an absorbing event is recurrent and cannot communicate with any event other than itself.

Example 6.2.4.1

The Markov Chain represented by the following matrix and graphed in figure 36 has two absorbing events, e_3 and e_4 ..

$$P_4 = \begin{pmatrix} \cdot 4 & \cdot 4 & \cdot 2 & 0 \\ \cdot 8 & 0 & \cdot 1 & \cdot 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

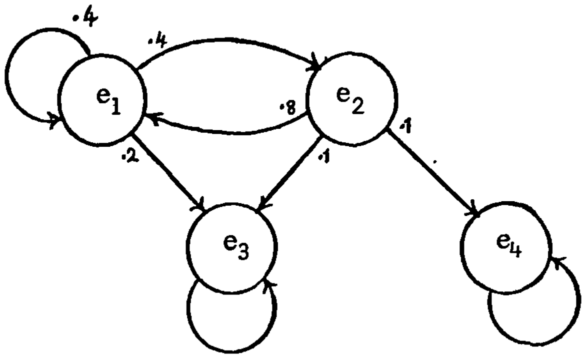


figure 36

A Markov Chain with absorbing events

Two possible realizations of event sequences are given:

1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2
1 1 3 3 3 3 3 3 3 3 3 3 3 3

and

1 1 1 2 1 1 1 1 1 1 2 1 1 2 4 4
4 4 4 4 4 4 4 4

With the event space defined in figure 37(a), a sonic mapping of these sequences is given in figures 37(b) and 37(c) (page 112).

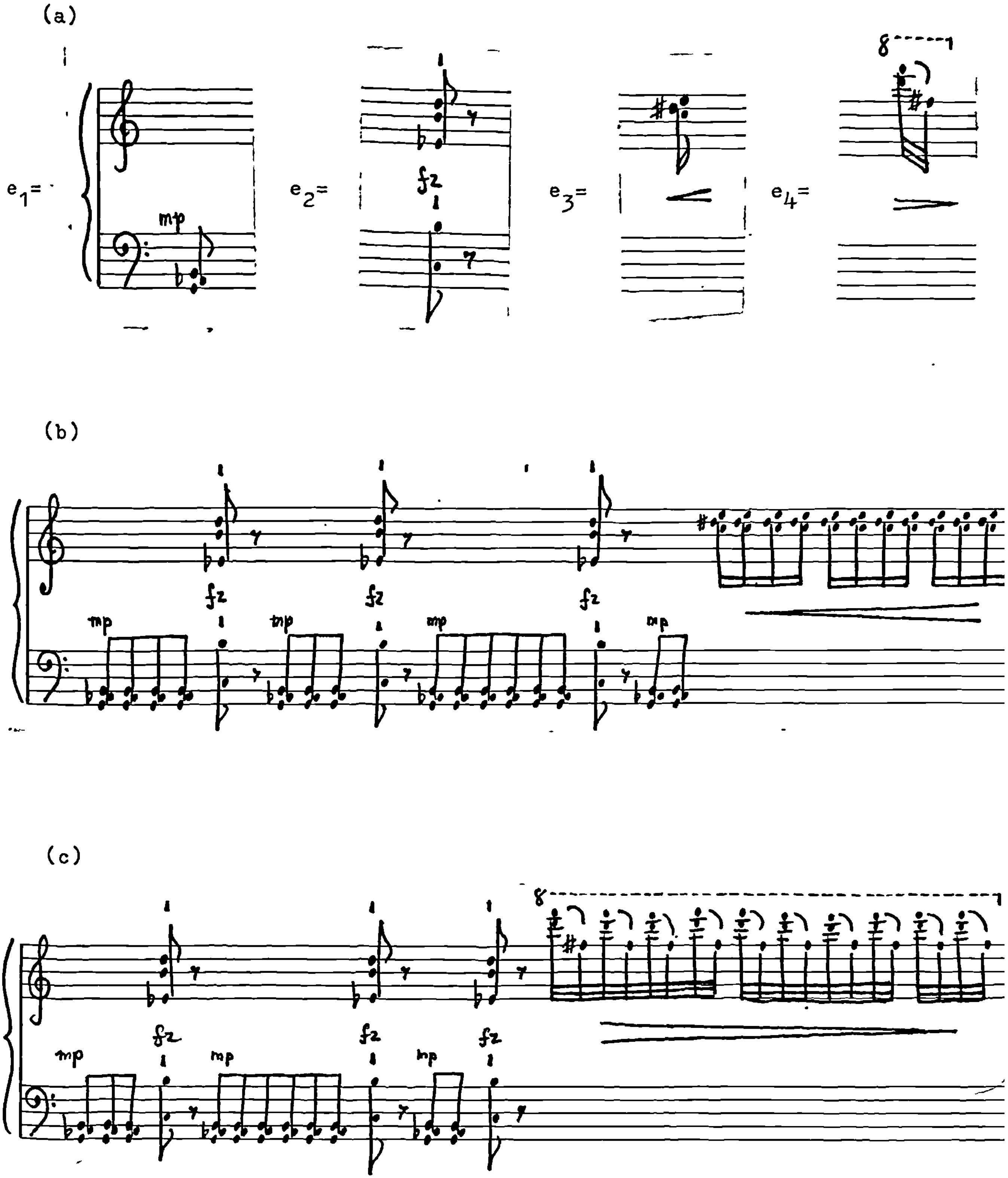


figure 37

An event space and two possible event sequences formed by applying the Markov Chain of figure 36

6.3 Classifying Markov Chains

Having defined and classified certain types of events, a consideration of the way in which these events are grouped together makes it possible to analyse and classify various types of Markov Chains.

6.3.1 Equivalence classes of events

It was stated in section 6.2.1 that the communication relation was an *equivalence relation*. As a result of this fact, all of those events in a Markov Chain which communicate with each other form an *equivalence class*. The results of sections 6.2.2 and 6.2.3, namely, that if two events communicate then they are either both recurrent or both transient, and that two communicating events always have the same period, lead to the general conclusion that *all the events in an equivalence class of events are either recurrent or transient, and all have the same period.*

This leads to two important results:

1. Once a process has left an equivalence class of *transient* events it cannot return to it.

2. An equivalence class C of *recurrent* events forms a *closed set* such that once a process has entered the set, it is impossible to move out of it, i.e.

$$p_{ij} = 0 \quad \text{if} \quad e_i \in C \quad \text{and} \quad e_j \notin C$$

An absorbing event is a closed set of order one containing just a single recurrent event.

Any Markov Chain can be split up into discreet equivalence classes of events. Such a partitioning is useful in describing the structure and behaviour of a Markov Chain, particularly where the number of events involved is large.

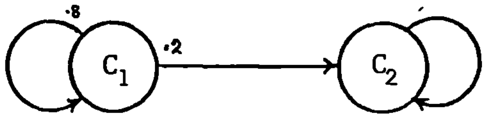
Example 6.3.1.1

Consider the Markov Chain defined by matrix P_3 in example 6.2.1.2 (figure 35, page 105).

This splits into two equivalence classes:

$$C_1 = \langle e_3 \rangle$$
$$C_2 = \langle e_1, e_2 \rangle$$

C_1 consists of just one transient event e_3. The closed set C_2 contains two recurrent events e_1 and e_2, both of period 2. Their relationship can be represented by the *class relation diagram* below.



Example 6.3.1.2

Consider the Markov Chain defined by matrix P_4 in example 6.2.4.1 (figure 36, page 111).

This consists of the following equivalence classes:

$$C_1 = \langle e_1, e_2 \rangle$$
$$C_2 = \langle e_3 \rangle$$
$$C_3 = \langle e_4 \rangle$$

C_1 contains two transient events e_1 and e_2, C_2 and C_3 are both closed sets each consisting of just one absorbing event. This Markov Chain has the class relation diagram graphed in figure 38.

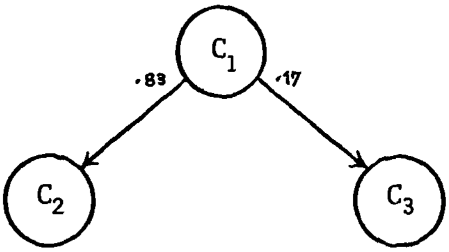


figure 38

Class-relation diagram of the Markov Chain of figure 35

The class transition probabilities were derived in the following way. Consider the events e_1 and e_2 in equivalence class C_1 and their transition probabilities while the process remains in this class. Isolating the rows and columns for just these two events in the transition matrix gives the following:

$$\begin{bmatrix} \cdot 4 & \cdot 4 \\ \cdot 8 & 0 \end{bmatrix}$$

This is not a stochastic matrix, but adjusting the probabilities gives the following result, graphed in figure 39 (page 116).

$$\begin{bmatrix} \cdot 5 & \cdot 5 \\ 1 & 0 \end{bmatrix}$$

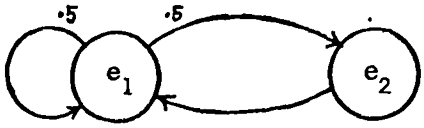


figure 39

Event-relation sub-diagram of events in equivalence class C_1 of figure 38

It can be seen that for every occurance of event e_2, event e_1 can be expected to occur twice, and so event e_1 will occur twice as often as event e_2. The overall distribution of events *while the process remains in this class* is thus given by the following probability assignment:

$$\Pi = \left(\tfrac{2}{3}, \tfrac{1}{3}\right) \quad \text{or} \quad (\cdot 67, \cdot 33)$$

This result can be derived by matrix multiplication, the use of which is considered later; but in this particular example the process is simple enough for the probabilities to be obvious.

It is apparent therefore that the process is likely to leave this class from event e_1 with a probability of $\frac{2}{3}$, in which case it ends up at class C_2, and to leave from event e_2 with a probability of $\frac{1}{3}$.

If the process leaves by event e_2, reference to the original matrix and diagram shows that it is equally probable to proceed to e_3 or e_4, so event e_3 will follow from e_2 with an overall probability of $\frac{1}{6}$.

Since C_2 can be approached in two ways, from e_1 with a probability of $\frac{2}{3}$ and from e_2 with a probability of $\frac{1}{6}$, these values are summed to give an overall probability of $\frac{5}{6}$.

Having thus derived the class transition probabilities it can be seen that the process is five times more likely to settle on event e_3 than on event e_4.

Example 6.3.1.3

A more complex Markov Chain of order eight is defined by the following matrix and graphed in figure 40 (page 118).

$$P_5 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdot 5 & \cdot 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdot 1 & \cdot 1 & \cdot 4 & \cdot 4 & 0 & 0 & 0 & 0 \\ 0 & \cdot 2 & 0 & 0 & \cdot 8 & 0 & 0 & 0 \\ 0 & 0 & \cdot 5 & \cdot 5 & 0 & 0 & 0 & 0 \\ \cdot 1 & 0 & \cdot 1 & 0 & \cdot 1 & \cdot 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot 4 & 0 & \cdot 3 & \cdot 3 \\ 0 & \cdot 2 & 0 & \cdot 2 & 0 & 0 & \cdot 6 & 0 \end{bmatrix}$$

The events have been numbered beginning with recurrent and ending with transient events. Notice that this tends to form a matrix consisting mainly of zeros above the diagonal. Such a matrix is said to be in *canonical form.*

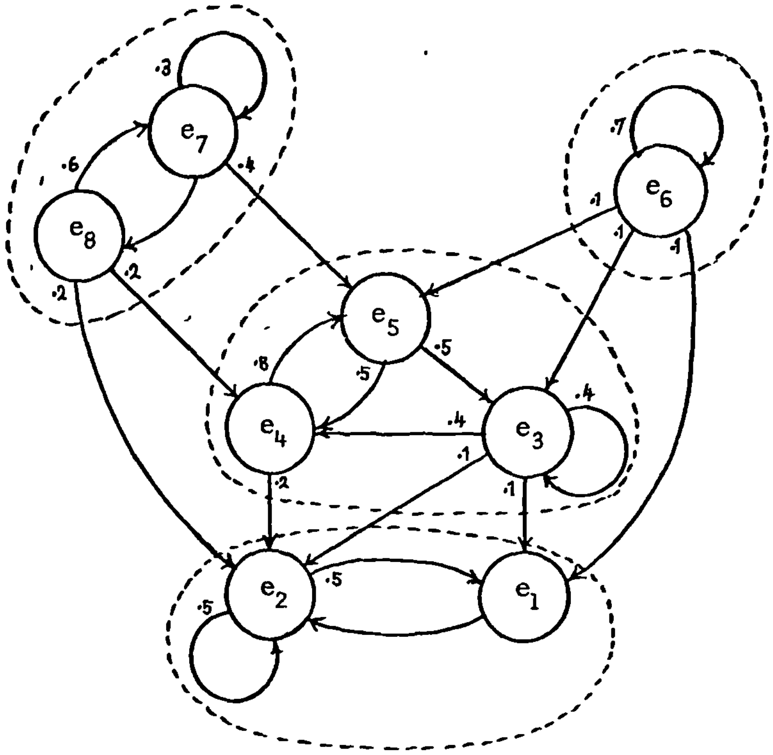


figure 40

Event-relation diagram of a Markov Chain of order eight consisting of transient classes and a recurrent class

Following the classifications developed above, it is possible to decompose this Markov Chain into the following equivalence classes:

$$C_1 = \langle e_6 \rangle$$

$$C_2 = \langle e_1 , e_2 \rangle$$

$$C_3 = \langle e_3 , e_4 , e_5 \rangle$$

$$C_4 = \langle e_7 , e_8 \rangle$$

C_1 , C_3 and C_4 are transient classes, C_2 is the only recurrent class. The class relation diagram of figure 41 can be constructed.

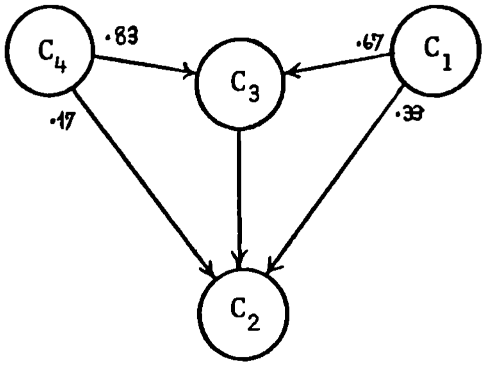


figure 41

Class-relation diagram of the Markov Chain in figure 40

It can be seen that this process will always settle on to the events in class C_2.

A number of possible sequences which might be generated from this Markov Chain are given:

$S_1 = e_6\ e_6\ e_3\ e_3\ e_4\ e_5\ e_3\ e_4\ e_2\ e_2\ e_1\ e_2\ e_1\ e_2\ e_1\ e_2\ e_2\ e_2$

$S_2 = e_6\ e_6\ e_6\ e_6\ e_1\ e_2\ e_1\ e_2\ e_1\ e_2\ e_2\ e_1\ e_2\ e_2\ e_1\ e_2\ e_1\ e_2$
$e_1\ e_2\ e_2$

$S_3 = e_8\ e_4\ e_2\ e_2\ e_1\ e_2\ e_1\ e_2\ e_2\ e_2\ e_2\ e_1\ e_2\ e_2\ e_1\ e_2\ e_1\ e_2$

$S_4 = e_7\ e_8\ e_7\ e_8\ e_4\ e_5\ e_4\ e_2\ e_2\ e_2\ e_2\ e_1\ e_2\ e_2\ e_2\ e_2\ e_1\ e_2$
$e_1\ e_2\ e_1\ e_2$

$S_5 = e_8\ e_7\ e_8\ e_4\ e_5\ e_4\ e_5\ e_4\ e_5\ e_3\ e_2\ e_2\ e_1\ e_2\ e_1\ e_2\ e_2\ e_1$
$e_2\ e_2\ e_2$

An event space is defined in figure 42 (page 121). For convenience the events have been grouped into their equivalence classes. The sequences have been transcribed in figure 43 (pages 122 - 123). When compared to the event relation diagram (figure 40) the underlying pattern structure of the sequences should be evident.

6.3.2 Irreducible Markov Chains

If a Markov Chain consists of only one equivalence class of communicating events then it is said to be *irreducible*. An irreducible Markov Chain can only consist of recurrent events. Of the examples considered so far, the Markov Chain defined by matrix P_1 of example 6.1.1 is irreducible.

There are a number of useful results which may be used to describe the constituent events of an irreducible Markov Chain. When a Markov Chain has been broken down into equivalence classes, further information about its structure may be gained by isolating each equivalence class and considering it as if it were an irreducible Markov Chain. This was done in example 6.3.1.2 above.

If an irreducible Markov Chain contains only ergodic events, then it is possible to derive a *limiting distribution* which defines the independent occurence probabilities of each event. In example 6.3.1.2 above, a limiting distribution was constructed for the sub-Markov Chain formed from class C_1. This distribution may also be derived by matrix multiplication.

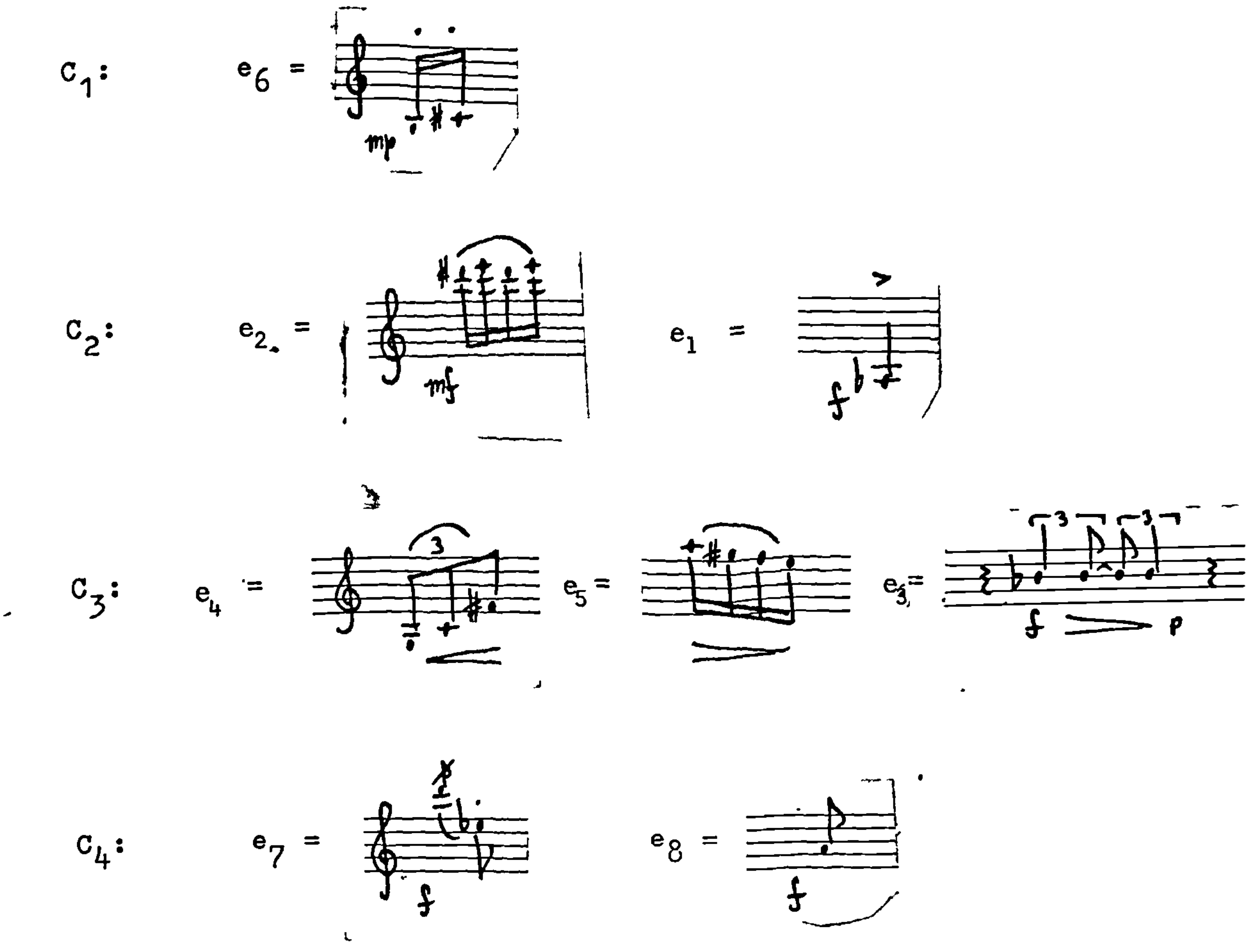


figure 42

An event space ~~for a~~ simple clarinet piece

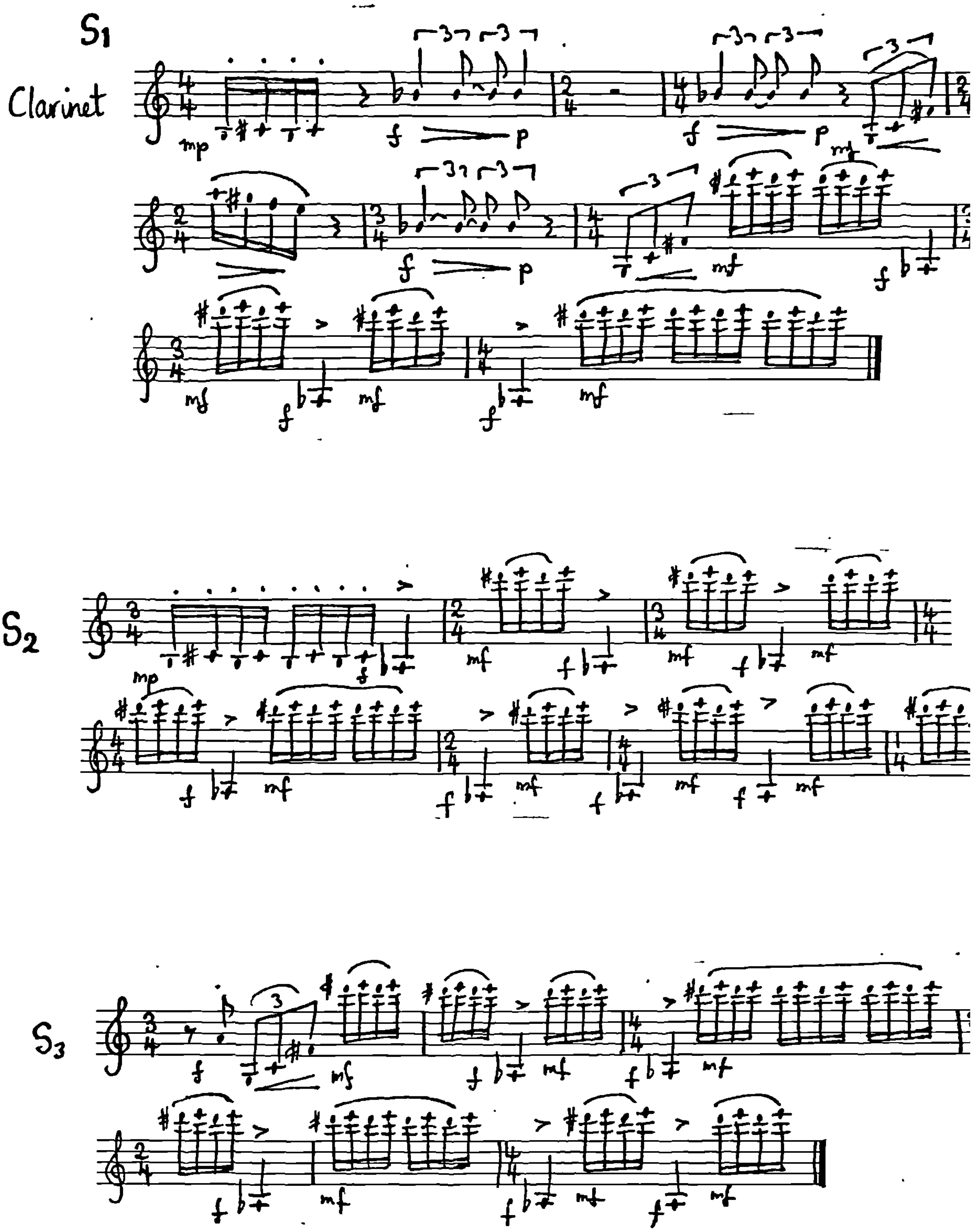
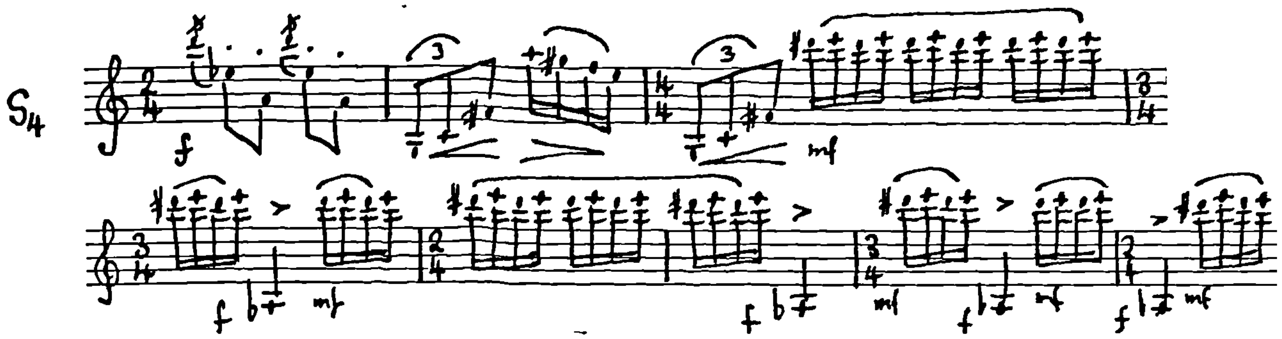


figure 43

Five possible event sequences formed by the application of the Markov Chain of figure 40 over the event space of figure 42



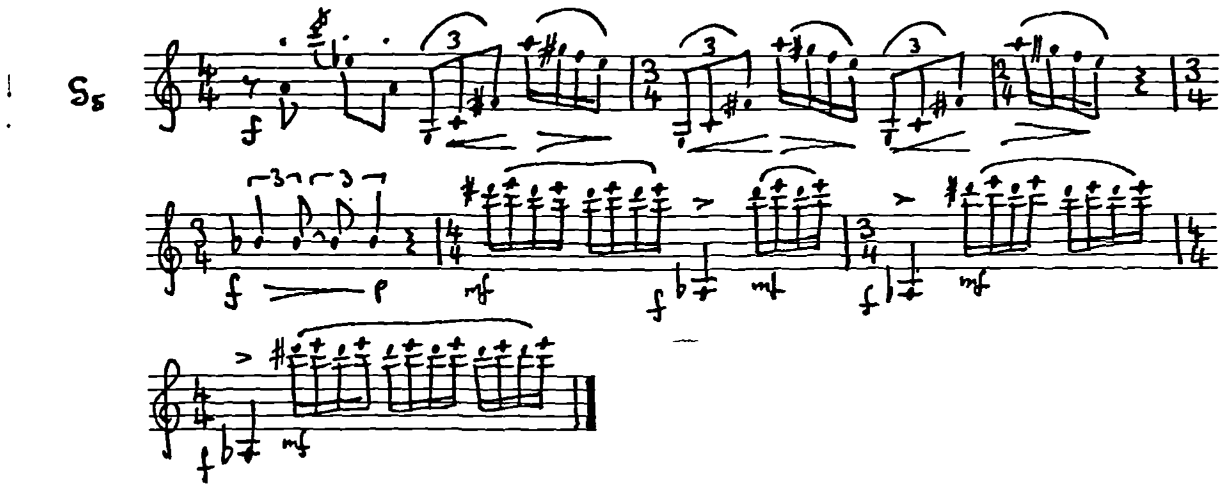


figure 43 continued

A complete mathematical explanation is beyond the scope of this thesis. The reader is referred to the appropriate mathematical texts for more detailed proofs if required.[3] A brief outline of the steps leading to this result is given.

If a matrix P of transition probabilities is multiplied by itself, the resulting matrix P^2 contains the probabilities that each event e_i will be followed by event e_j in two steps. If the matrix is raised to the n-th power, the resulting matrix P^n contains the probabilities $p_{ij}^{(n)}$ that event e_i will be followed by event e_j in n steps. As the value of n increases, the probabilities $p_{ij}^{(n)}$ in each matrix will eventually converge to a limit p_{ij}^* . Each of the i rows in this limiting matrix P^* will be identical. The entries in each row thus form a probability assignment Π which is known as the *limiting distribution.* Each entry π_j in the assignment gives the independent occurence probability of each event e_j in the Markov Chain.

Example 6.3.2.1

The limiting distribution of the sub-Markov Chain formed from equivalence class C_1 in example 6.3.1.2 above, was determined intuitively. This same distribution can now be derived formally from the stochastic matrix.

$$P_C = \begin{bmatrix} {\cdot}5 & {\cdot}5 \\ 1 & 0 \end{bmatrix}$$

$$P_C^2 = \begin{bmatrix} {\cdot}5 & {\cdot}5 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} {\cdot}5 & {\cdot}5 \\ 1 & 0 \end{bmatrix} = \left[\begin{matrix} ({\cdot}5)({\cdot}5) + ({\cdot}5)\,1 & ({\cdot}5)({\cdot}5) + (\\ 1\,({\cdot}5) + 0 & 1\,({\cdot}5) + 0 \end{matrix}\right.$$

$$\begin{bmatrix} {\cdot}75 & {\cdot}25 \\ {\cdot}5 & {\cdot}5 \end{bmatrix}$$

$$P_C^4 = (P_C^2)^2 = \begin{bmatrix} \cdot 75 & \cdot 25 \\ \cdot 5 & \cdot 5 \end{bmatrix} \begin{bmatrix} \cdot 75 & \cdot 25 \\ \cdot 5 & \cdot 5 \end{bmatrix} = \begin{bmatrix} \cdot 69 & \cdot 31 \\ \cdot 62 & \cdot 38 \end{bmatrix}$$

$$P_C^8 = (P_C^4)^2 = \begin{bmatrix} \cdot 67 & \cdot 33 \\ \cdot 67 & \cdot 33 \end{bmatrix} = P_C^*$$

$$\pi_1 = p_{11}^* = p_{21}^* = \cdot 67$$

$$\pi_2 = p_{12}^* = p_{22}^* = \cdot 33$$

$$\Pi = (\cdot 67 , \cdot 33)$$

It can be seen that the matrix, under multiplication, quickly converges to the limiting distribution P_C^* .

It is possible to derive from the limiting distribution the *mean recurrence time* μ_j for each event e_j in the Markov Chain. The mean recurrence time is the average number of steps expected in a sequence before event e_j recurs. It is simply the inverse of the event's limiting probability:

$$\mu_j = \frac{1}{\pi_j}$$

Example 6.3.2.1 continued

$$\mu_1 = \frac{1}{\cdot 67} = 1\cdot 5 \qquad \mu_2 = \frac{1}{\cdot 33} = 3$$

So event e_2 can be expected to recur after three steps and event e_1 after one and a half steps, on average.

If an irreducible Markov Chain does not contain ergodic events, in other words it has a period greater than one, then it is not possible to obtain the limiting distribution in this way.

Example 6.3.2.2

Consider the sub-Markov Chain formed by events e_1 and e_2 from the matrix P_3 defined in example 6.2.1.2 (page 105). They both have a period of two. The sub-stochastic matrix is:

$$P_{3C} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

If this matrix is squared, the identity matrix is obtained:

$$P_{3C}^{2} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Further multiplication will yield only P_{3C} and I alternately; so this matrix never converges to a limit.

6.3.3 Ergodic Markov Chains

If the equivalence classes of a Markov Chain include just one recurrent class consisting of ergodic events, then it is said to be an *ergodic Markov Chain*. The behaviour of such a system is perfectly predictable. In the Markov Chain of example 6.3.1.3 (page 118) it can be seen that the process will always move eventually into equivalence class C_2. This chain is ergodic. With the Markov Chain of example 6.2.4 (page 110), which has two recurrent classes $\langle e_3 \rangle$ and $\langle e_4 \rangle$, the process may enter either of the absorbing events e_3 or e_4 , it is not certain which. The behaviour of such a system is not predictable. This chain is not ergodic.

The limiting distribution of an ergodic Markov Chain corresponds to the limiting distribution of its recurrent class. As the number of steps increases, the significance of the events in the transient classes lessens and so their overall independent probabilities tend towards zero. Thus the limiting distribution of the Markov Chain P_5 of example 6.3.1.3 is the same as that of its recurrent class C_2. It can be seen that the sub-Markov Chain formed by this class is the same as the sub-Markov Chain considered in example 6.3.2.1 above, but with the events numbered differently. They therefore have a similar limiting distribution:

$$\Pi_{(P_5)} = (\cdot 33 , \cdot 67)$$

$$\Pi_{(P_4/C_1)} = (\cdot 67 , \cdot 33)$$

Most of the musical examples which have been considered so far have been in terms of notes expressed in conventional notation. However, it has been pointed out that the structuring techniques being described are equally applicable at the micro-level; an application which is particularl) explored in the following chapter. Ergodic Markov Chains can be particularly useful in defining sounds which begin with noisy or complex transient characteristics and then settle down into a more regular oscillation. This is the case with all instrumental sounds to a small extent, but the pattern is most particularly apparent with many percussive instruments such as cymbals, drums and bells. Such sounds will never begin in a precisely identical fashion and undergo continuous subtle forms of minor modification throughout their whole duration. Using a Markov Chain to model such sound structures on a computer music system gives the result a lively vigorous quality associated with natural sounds, but always within a consistent and controllable framework.

If the Markov Chain is non-ergodic it will either contain periodic events or more than one recurrent class. In the former case, if used for sound synthesis, the regular cycles will produce an artificially consistent sound, characteristic of 'electronic' music. In the latter case, the outcomes will be unpredictable and results obtained which are inconsistent with each other. Non-ergodic Markov Chains can be useful to produce certain effects or if differing degrees of control are sought in specific composing situations, but are inappropriate for the application suggested above.

6.3.4 Stationary Markov Chains

A Markov Chain with all its rows equal, i.e.:

$$p_{1j} = p_{2j} = \ldots = p_{nj} \qquad j = 1, \ldots, n$$

is known as a *Stationary Markov Chain.*

It can be seen that in such a case, the probabilities of each event's occurence are *independent*, for no matter what event e_i precedes event

e_j its occurence probability will always be the same.

Since the rows, and hence the event distributions, are identical from the start, the limiting matrix of a stationary Markov Chain will be identical to the original matrix:

$$p^*_{ij} \quad = \quad p_{ij} \qquad \text{for all} \quad i , j = 1 , \ldots , n$$

$$\text{and so} \qquad P^* \quad = \quad P \qquad .$$

This can easily be verified by simple matrix multiplication.

A staionary Markov Chain is identical to the simple probability distributions discussed in chapter four. It can thus be seen that such probability distributions are merely a special case of the more general system of Markov Chains.

It will be shown later that the Markov Chains which have been descibed can themselves be considered as a special case of more general systems: Multi-dimensional Markov Chains, Markov Chains with event vectors and Markov Chains constructed from events which are themselves defined as Markov Chains. These will be considered later in this chapter. Formal grammars, of which Markov Chains can also be considered to be a special case, are developed in chapter nine.

First of all some particular event patterns in Markov Chains will be identified.

6.4 Some event patterns in Markov Chains

6.4.1 Random Walk

If the probabilities in a Markov Chain are such that:

$$p_{11} = p_{nn} = 1$$

$$p_{i,i+1} = q \ , \quad p_{i,i-1} = 1 - q \qquad \text{for } i = 2, 3, \ldots, n-$$

$$\text{and} \quad p_{ij} = 0 \quad \text{otherwise,}$$

so that the matrix has the following form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ 1-q & 0 & q & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ 0 & 1-q & 0 & q & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 1-q & 0 & q \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 & 1 \end{bmatrix}.$$

then the process it describes is called a *random walk.* If the process begins with event e_i, then it will move up or down to event e_{i+1} or e_{i-1}, always to an adjacent event. Eventually it will reach event e_1 or e_n at which point it stops, for events e_1 and e_n are both absorbing events. Such a process is therefore particularly described as a *random walk with absorbing barriers.*

If the probabilities in the chain are such that:

$$p_{12} = p_{n,n-1} = 1$$

$$p_{i,i+1} = q \; , \quad p_{i,i-1} = 1-q \quad \text{for} \quad i = 2, 3, \ldots, n-1$$

and $\quad p_{ij} = 0 \quad$ otherwise,

so that the matrix has the following form:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ 1-q & 0 & q & 0 & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ 0 & 1-q & 0 & q & \cdot & \cdot & \cdot & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 1-q & 0 & q \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 & 1 & 0 \end{bmatrix}$$

then once the process reaches event e_1 or e_n it will simply turn round and work its way back again. Such a process is known as a *random walk with reflecting barriers.*

Example 6.4.1.1

A random walk with absorbing barriers is defined on an event space of order twelve, and has the following matrix representation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdot 6 & 0 & \cdot 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = P_6$$

Suppose that the process begins with event e_9. Figure 44 plots in graphical form what might happen as the process moves from event to event:

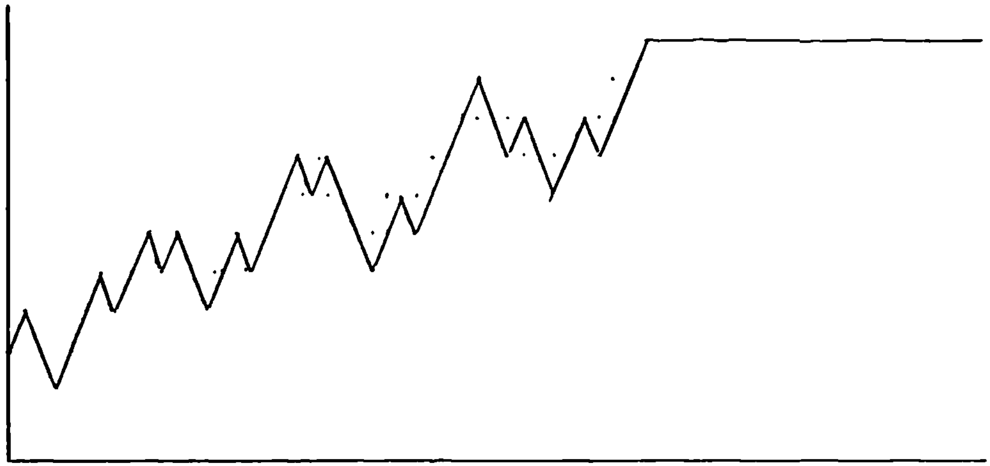
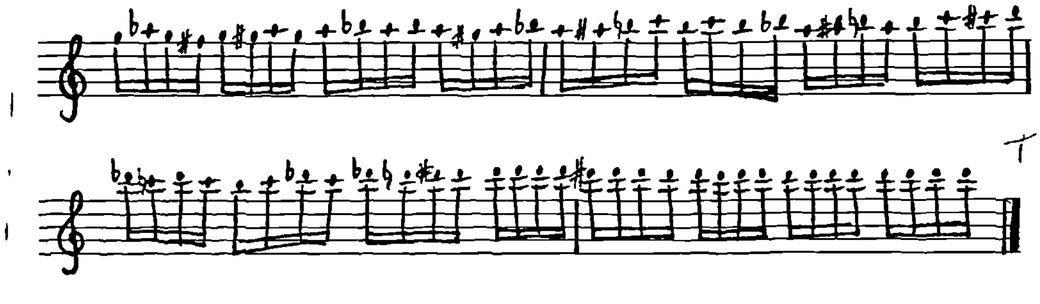


figure 44

Representation of an event sequence generated by a Markov Chain with absorbing barriers

The process gradually rises to event e_1 and then stays there. When applied to an event space consisting of notes in the chromatic scale above E' , the result of figure 45 is obtained.



Musical transcription of figure 44

figure 45

Exchanging the probabilities, i.e. making $p_{i,i-1} = \cdot 4$ and $p_{i,i+1} = \cdot 6$, will produce an opposite result where the process gradually falls to settle on event e_{12}.

Example 6.4.1.2

On the same event space, a random walk with reflecting barriers is defined by the following matrix:

$$\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdot 5 & 0 & \cdot 5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix} = P_7$$

Figure 46 represents a possible plot of this process.

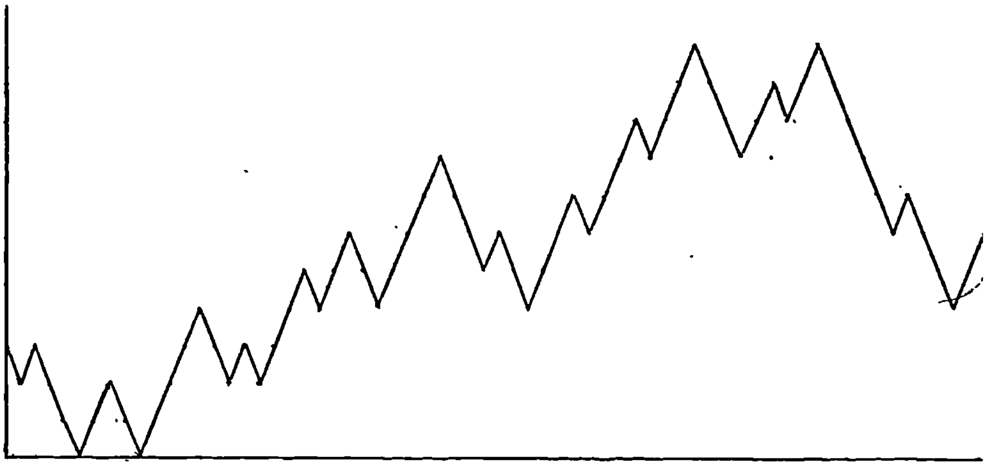


figure 46

Representation of an event sequence generated by a Markov Chain with reflecting barriers

Using the same musical interpretation.as the previous example gives figure 47.

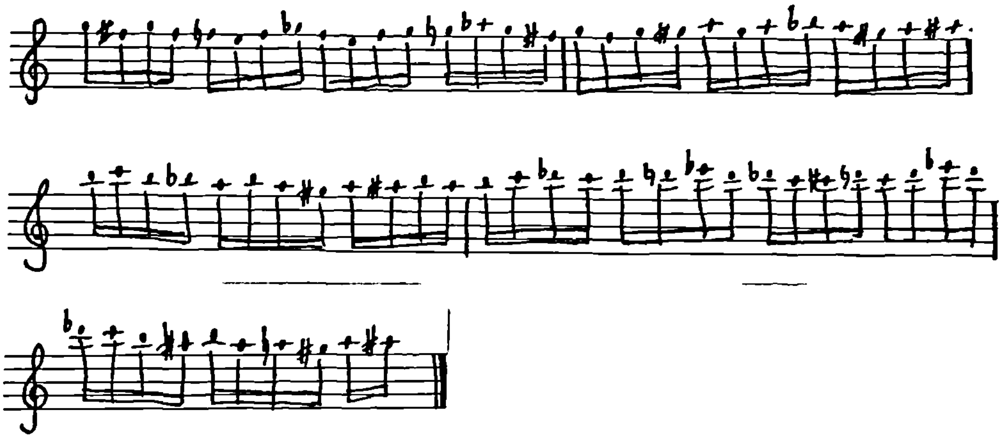


figure 47

Musical transcription of figure 46

The random walk process was applied at a textural or micro-level to produce particular effects in the second movement of *MACRICISUM*. Their use is described in section 7.3 .

The random walk model has been used to describe certain types of movement in the physical world, in particular the "Brownian Motion" of a particle subject to a large number of molecular collisions. The 'curves' drawn above (figures 44 and 46) belong to a particular class of mathematical functions known as *fractals* which have recently been identified and studied by Mandelbrot.[4] They have the property that the same basic shape structure is maintained whether considered in close-up or from a distance. It can be seen that they have a certain resemblance to the profile of a mountain range, and indeed formal studies have shown that a mountain profile can be modelled using the random walk notion. Richard Voss has produced simple computer-generated landscapes in this way. He has also used the same model to generate simple melodies which he terms *Brown Music* ! [5]

6.4.2 Difference distributions and Markov Chains

The difference distributions introduced in section 4.7 have, in their simplest form, precisely the same structure as a random walk with reflecting barriers. In this case the difference space will consist just of (-1,+1). The pattern can be extended over a larger difference space so that each row in the corresponding stochastic matrix contains the same probabilities, but they will be shifted one place to the right in each successive row.

Thus if the difference space has limits $(-\ell, +m)$, the entries in each row of the matrix will have the following form:

$$p_{i,i-\ell},\ p_{i,i-\ell+1},\ \ldots,\ p_{i,i-1},\ p_{i,i},\ p_{i,i+1},\ \ldots,\ p_{i,i+m}$$

where, $p_{i,i-\ell} > 0$, $p_{i,i+m} > 0$

and $$\sum_{k=1}^{\ell} p_{i,i-k} + \sum_{k=0}^{m} p_{i,i+k} = 1$$

Example 6.4.2.1

Consider the difference space D_E of example 4.7.1 (page 81):

$$D_E = (-4, -3, -2, -1, 0, +1, +2)$$

which was defined over an event space of order six and has the following probability assignment:

$$P_{D_E} \quad = \quad (\cdot 2 , 0 , 0 , \cdot 3 , \cdot 1 , \cdot 2 , \cdot 2)$$

The corresponding stochastic matrix may be constructed by writing the difference space probability assignment in each row, with the entry corresponding to the 0 difference in the difference space (in this case $\cdot 1$, in the fifth position) always centering on p_{ii}, that is on the main diagonal of the matrix. Where there are too many entries to fit in to the left or right of p_{ii} they are left out, and the probabilities adjusted afterwards to make each row total unity. Thus the intermediary matrix is as below:

$$\begin{pmatrix} \cdot 1 & \cdot 2 & \cdot 2 & 0 & 0 & 0 \\ \cdot 3 & \cdot 1 & \cdot 2 & \cdot 2 & 0 & 0 \\ 0 & \cdot 3 & \cdot 1 & \cdot 2 & \cdot 2 & 0 \\ 0 & 0 & \cdot 3 & \cdot 1 & \cdot 2 & \cdot 2 \\ \cdot 2 & 0 & 0 & \cdot 3 & \cdot 1 & \cdot 2 \\ 0 & \cdot 2 & 0 & 0 & \cdot 3 & \cdot 1 \end{pmatrix}$$

which with adjusted probabilities becomes the stochastic matrix P_8:

$$\begin{pmatrix} \cdot 2 & \cdot 4 & \cdot 4 & 0 & 0 & 0 \\ \cdot 375 & \cdot 125 & \cdot 25 & \cdot 25 & 0 & 0 \\ 0 & \cdot 375 & \cdot 125 & \cdot 25 & \cdot 25 & 0 \\ 0 & 0 & \cdot 375 & \cdot 125 & \cdot 25 & \cdot 25 \\ \cdot 25 & 0 & 0 & \cdot 375 & \cdot 125 & \cdot 25 \\ 0 & \cdot 33 & 0 & 0 & \cdot 5 & \cdot 17 \end{pmatrix}$$

The difference distibution used in the example above could be applied to any event space of order greater than or equal to five, since the maximum difference is -4 . In general, it can be seen that if the order of the event space E is n, then the maximum difference will be $n-1$. Thus the maximum size of the difference space D_E will be the sum of $2(n-1)$, for positive and negative differences, and 1 for a difference of 0, i.e.

$$|D_E| \leq 2(n-1) + 1 = 2n - 1 \quad .$$

It can be seen that the matrix representation P_8 of the difference distribution above is much more complex than the original difference space definition D_E and its corresponding probability assignment P_{D_E}. It is therefore much more efficient to use the difference distribution representation for a pattern structure whenever the absolute values of events are not important, but only the difference relation between them. This is obviously only true when the events in an event space fall naturally into a regular, related sequence.

In a musical context this is very often the case. For example it is frequently only the intervals between notes which are significant rather than their absolute pitches. If a piece is transposed into a different key, or if the whole pitch is shifted, and when an instrument is used which has been tuned to a different standard (<u>not</u> a different temperament!), the pitch relations remain identical and the piece sounds the same.

6.4.3 Alternating events

If for a given event e_k:

$p_{kj} > 0$ for $k \neq j$, $j = 1 , \ldots , k-1 , k+1 , \ldots , n$

$p_{ik} = 1$ for $i \neq k$, $i = 1 , \ldots , k-1 , k+1 , \ldots , n$

and $p_{ij} = 0$ otherwise, in particular $p_{kk} = 0$

then the resulting sequence will consist of event e_k alternating with the other events in the event space. The matrix will have the form:

$$\begin{pmatrix} 0 & 0 & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \alpha_1 & \alpha_2 & \cdot & \cdot & \cdot & \alpha_{k-1} & 0 & \alpha_{k+1} & \cdot & \cdot & \cdot & \alpha_{n-1} & \alpha_n \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 \end{pmatrix}$$

$$\sum_{j=1}^{n} \alpha_j = 1$$

In other words the entries will be in the form of a cross. This can be represented by figure 48.

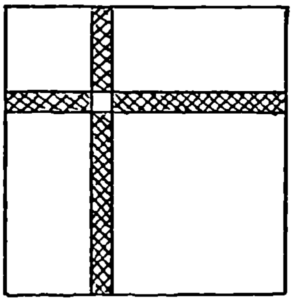


figure 48

General form of the matrix representing a Markov Chain with one alternating event

If the events are arranged so that the alternating event is e_1, the following simplified matrix will ensue:

$$\begin{pmatrix} 0 & \alpha_1 & \alpha_2 & \cdot & \cdot & \cdot & \alpha_n \\ 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \end{pmatrix}$$

and the diagram will appear as figure 49.

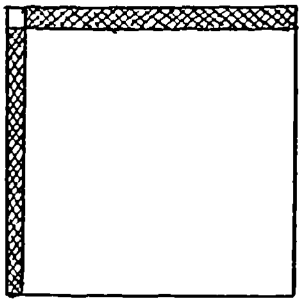


figure 49

Simplified form of the matrix representing a Markov Chain with one alternating event

Such a Markov Chain is irreducible, and periodic with a period of two. Example 6.1.2 (page 97) is a specific instance of such a distribution. (See also figures 32 and 33).

If $\alpha_i = \alpha_j$ $(i , j = 1 , \ldots , n)$ then the other events will all alternate with equal probability. If, in general, $\alpha_i \neq \alpha_j$ then the distribution will be biased towards certain events.

If an attempt is made to draw the event relation diagram of the general simple alternating process, the result will look like a many-petalled flower with the alternating event at the centre (figure 50):

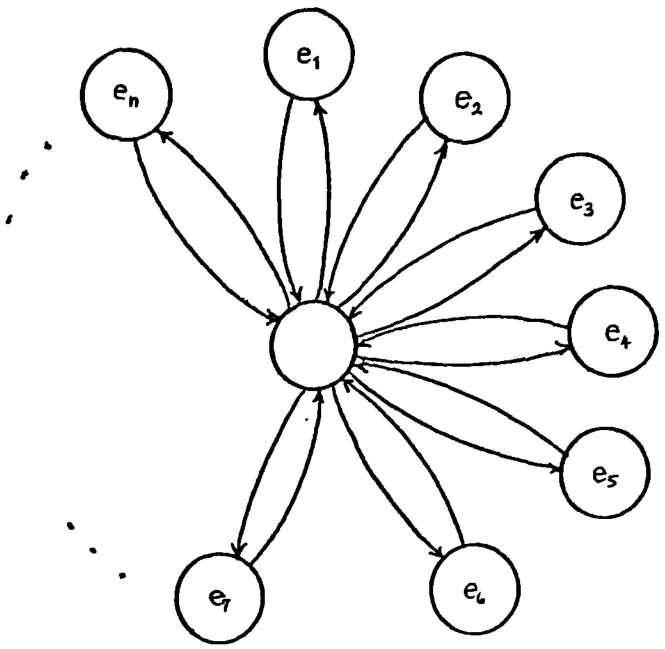


figure 50

Event relation diagram of the general simple alternating Markov Chain

This simple alternating process may be extended to have two, three or more alternating events which do not *directly* communicate with each other but alternate with the remaining events. In general it is possible to split the event space into two alternating classes. (N.B. These are not the same as the equivalence classes considered in section 6.3.1 above). Again it is most convenient to list the events such that members of one class appear first in the event space followed by the members of the other class. The matrix will have the form of figure 51.

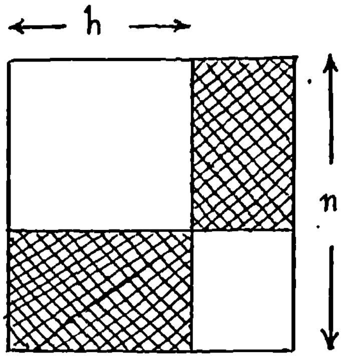


figure 51

General form of the matrix representing a Markov Chain with two alternating classes of events

If there are h events in the first alternating class, and (n - h) in the second then:

$p_{ij} > 0$ when both $i > h$ and $j \leq h$

and both $i \leq h$ and $j > h$

$p_{ij} = 0$ when both $i \leq h$ and $j \leq h$

and both $i > h$ and $j > h$.

The event relation diagram of the general two-class alternating process is rather fearsome, like the jaws of a gum-chewing shark! (figure 52).

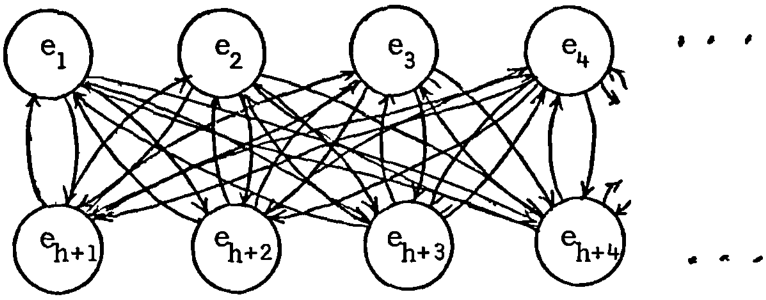


figure 52

Event relation diagram of the general two-class alternating Markov Chain

A simple musical example illustrating this process is now given.

Example 6.4.3.1

The following matrix defines a two-class alternating Markov Chain over an event space of order ten.

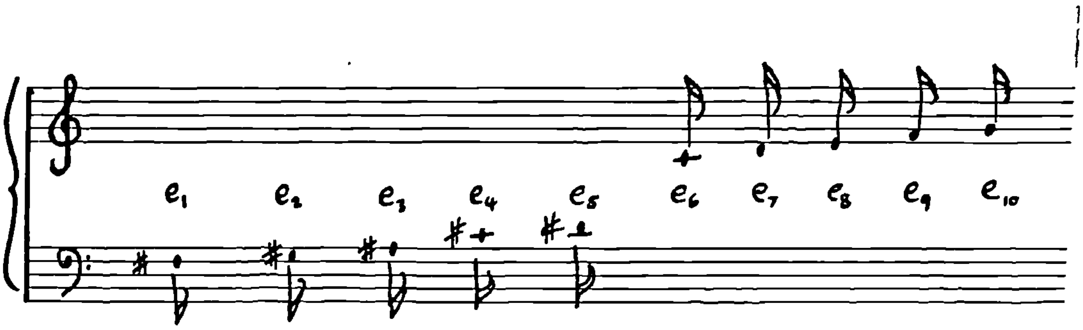
$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & \cdot 1 & \cdot 1 & \cdot 2 & \cdot 3 & \cdot 3 \\
0 & 0 & 0 & 0 & 0 & \cdot 3 & \cdot 3 & \cdot 2 & \cdot 1 & \cdot 1 \\
0 & 0 & 0 & 0 & 0 & \cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 \\
0 & 0 & 0 & 0 & 0 & \cdot 2 & \cdot 4 & \cdot 1 & \cdot 1 & \cdot 2 \\
0 & 0 & 0 & 0 & 0 & \cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 \\
\cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 & 0 & 0 & 0 & 0 & 0 \\
\cdot 2 & \cdot 1 & \cdot 1 & \cdot 4 & \cdot 2 & 0 & 0 & 0 & 0 & 0 \\
\cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 & \cdot 2 & 0 & 0 & 0 & 0 & 0 \\
\cdot 1 & \cdot 1 & \cdot 2 & \cdot 3 & \cdot 3 & 0 & 0 & 0 & 0 & 0 \\
\cdot 3 & \cdot 3 & \cdot 2 & \cdot 1 & \cdot 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

This may generate the following event sequence:

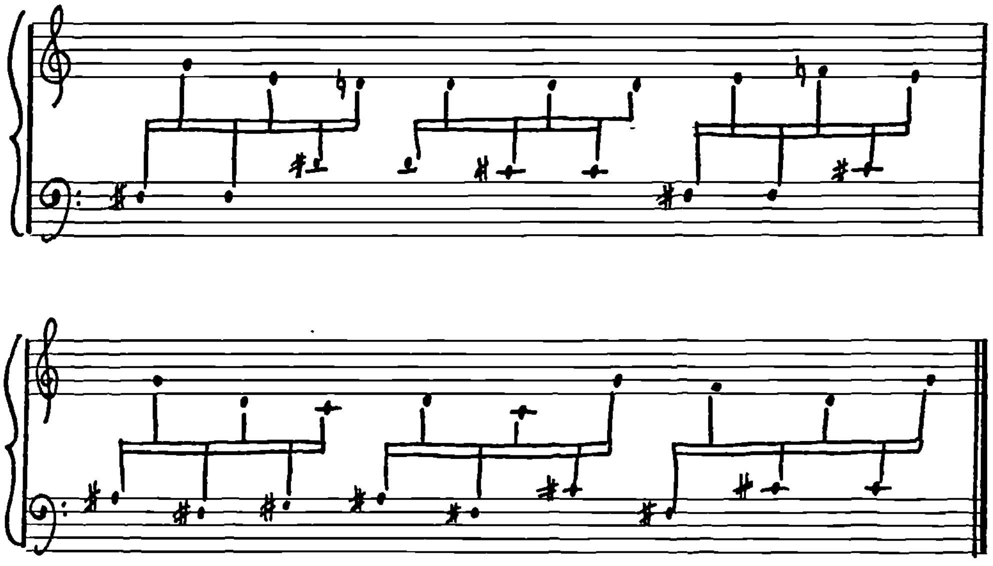
$$\begin{array}{cccccccccccc}
e_1 & e_{10} & e_1 & e_8 & e_5 & e_7 & e_5 & e_7 & e_4 & e_7 & e_4 & e_7 \\
e_1 & e_8 & e_1 & e_9 & e_4 & e_8 & e_3 & e_{10} & e_1 & e_7 & e_2 & e_6 \\
e_3 & e_7 & e_1 & e_6 & e_4 & e_{10} & e_1 & e_9 & e_4 & e_7 & e_4 & e_{10}
\end{array}$$

Using the event space defined in figure 53(a), the sequence will transcribe into figure 53(b) (page 143). This has rather an interesting similarity to some of Debussy's piano music.[6]

The two-class alternating process can be further extended to an m-class alternating process. If M is a sub-stochastic matrix associated with an alternating class, then the m matrices must be distributed within the lattice of the main stochastic matrix in such a way that only one M appears in each row or column of the lattice, otherwise absorbing classes will ensue, and the remaining events will fall into transient classes.



(a)



(b)

figure 53

Event space with a sequence generated with a two-class alternating Markov Chain

Thus if M_{ij} is the matrix in the i-th row and j-th column of the stochastic matrix lattice, entries will appear at M_{12}, M_{23}, ... , $M_{m-1,m}$ and at $M_{m,1}$ and the matrix will have the form of figure 54.

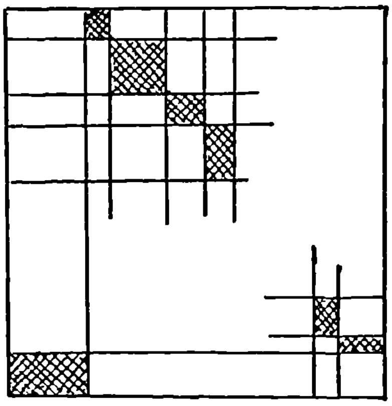


figure 54

General form of the matrix representing an m-class alternating Markov Chain

The event relation diagram of the general m-class alternating Markov Chain looks like the clothoid's web of fate! (figure 55). There are no connections along the axes, each of which represents one alternating class. It is only possible to return to any event after completing the cycle by visiting each class in turn. An m-class alternating Markov Chain is periodic with a period equal to m .

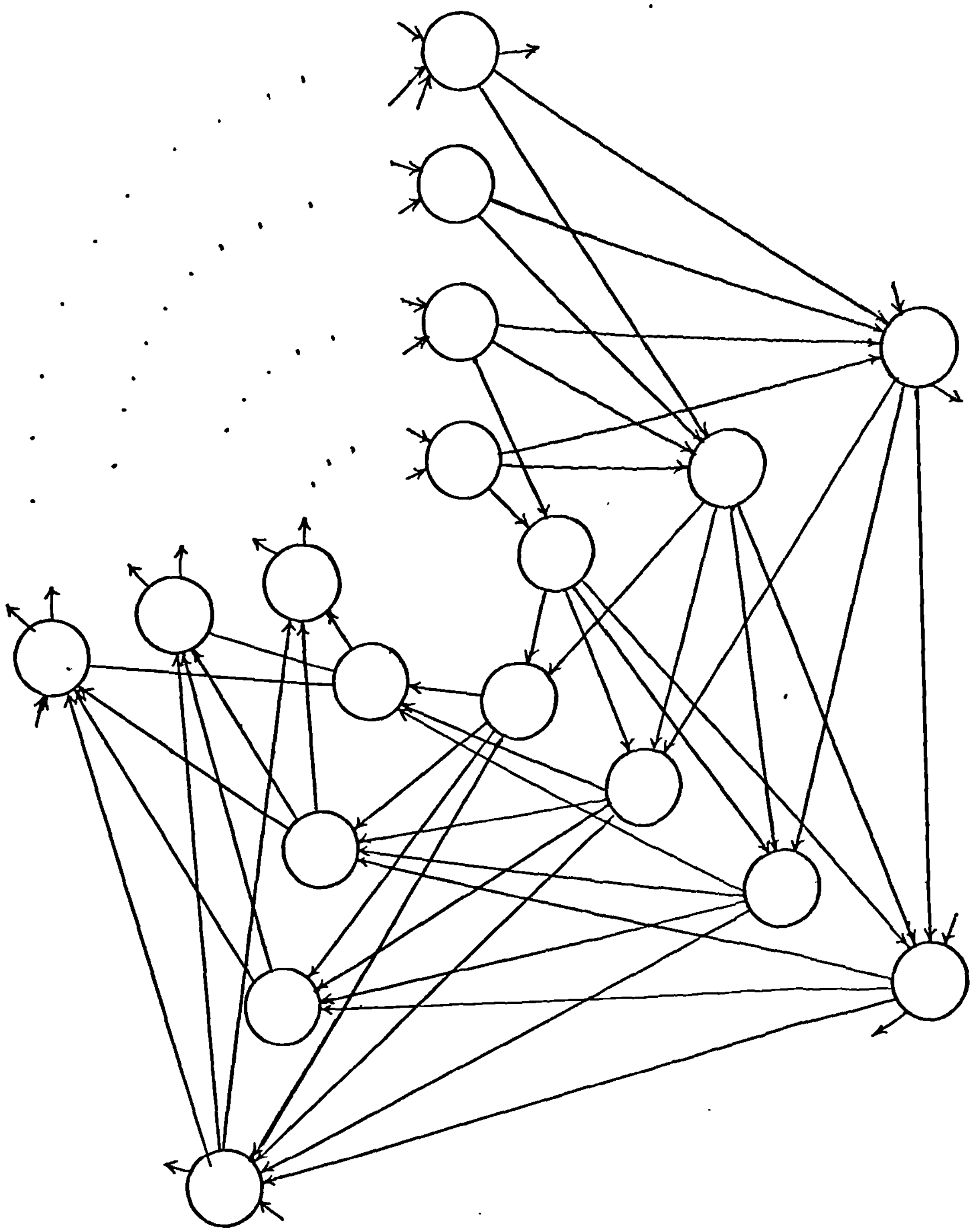


figure 55

The event relation diagram of a general m-class alternating Markov Chain

6.4.4 An expanding range of events.

This process is essentially a generalisation of the random walk notion where the range of possible events increases with each row of the matrix.

ν_i will be used to represent the number of non-zero entries in the i-th row of the matrix.

If $\nu_i = i+1$ then:

$$p_{ij} > 0 \qquad j = 1 , \ldots , i+1$$
$$p_{ij} = 0 \qquad j = i+2 , \ldots , n$$

and the range of events available gradually expands.

The matrix will have the form of figure 56.

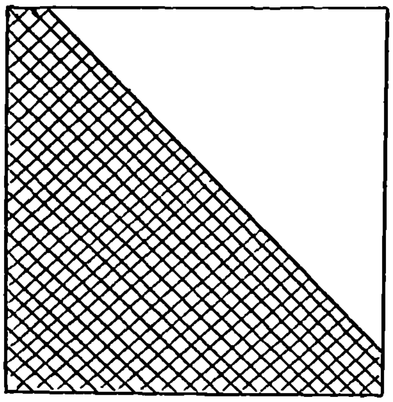


figure 56

General form of the matrix of a Markov Chain with an expanding range of events

Again the events have been ordered so that event e_1 is the starting event, and the other events expand towards event e_n. Since all events communicate, such a matrix is irreducible with all states recurrent. The following diagram illustrates the expected character of a resulting event sequence.

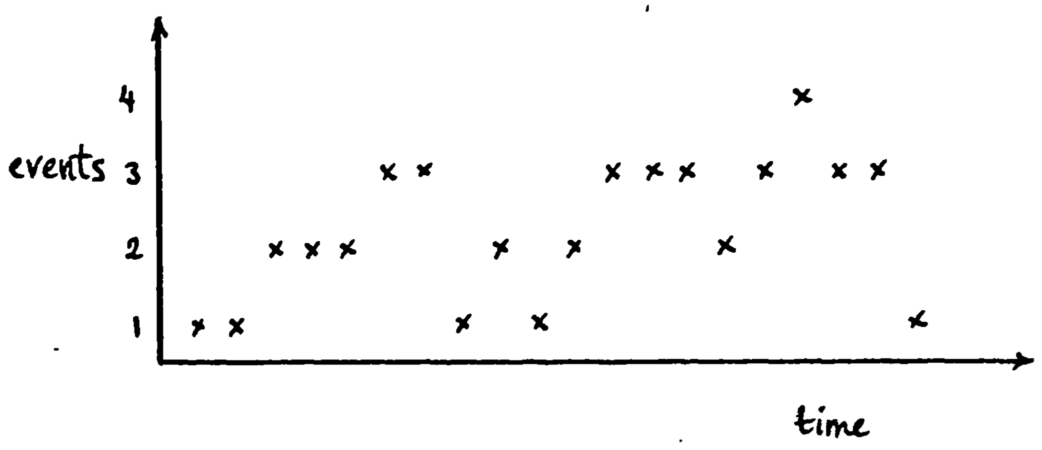


figure 57

A representation of a typical event sequence formed from a Markov Chain with an expanding range of events

6.4.5 A contracting range of events

It is not possible to have a general contracting range of events where the complete range narrows down to one event, for either:

(1) e_i can only be followed by e_{i-x} $(x \geq 0)$ in which case the process merely moves by steps to the final event (assumed to be e_1).

or (2) There is a possibility that e_i can be followed by e_{i+x} in which case the events will continue to move backwards and forwards, all events will communicate and the range will not contract.

However, a quasi-contracting range of events can be achieved by constructing a 'staircase' of transient classes with the recurrent event or event class at the summit.

The matrix, in canonical form (cf. example 6.3.1.3 page 117) will have the structure of figure 58.

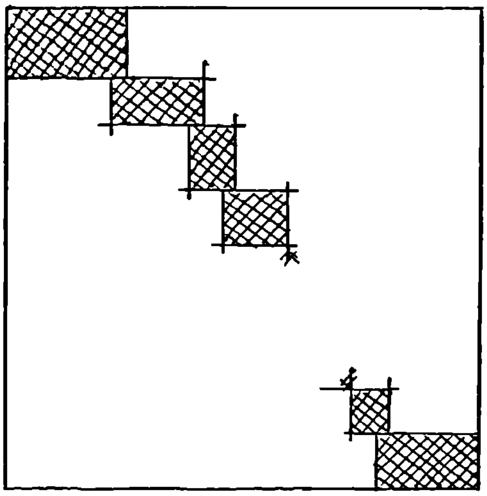


figure 58

General form of the matrix of a Markov Chain with a quasi-contracting range of events

Figure 59 (page 149) represents the sort of sequence that might be produced. In practice, the event space need not have its elements ordered consecutively. Thus it is possible to renumber the events in the event space of figure 59 to produce the form of figure 60 (page 149), where an effect closer to a general contrating system is obtained.

The structures produced by these Markov Chains are similar to the *Tendency* masks used in Koenig's SSP and other programs,[7] and in Truax's POD series.[8] In these other systems a simple regular density of events within the overall tendency mask limits is prescribed. However, more

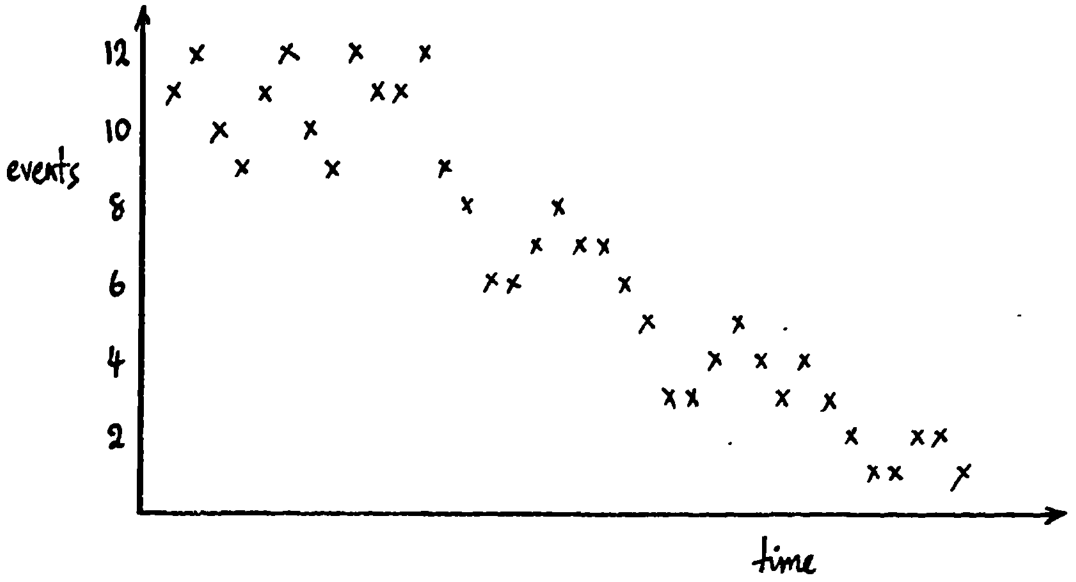


figure 59

Representation of a typical event sequence formed, from a Markov Chain of the type shown in figure 58

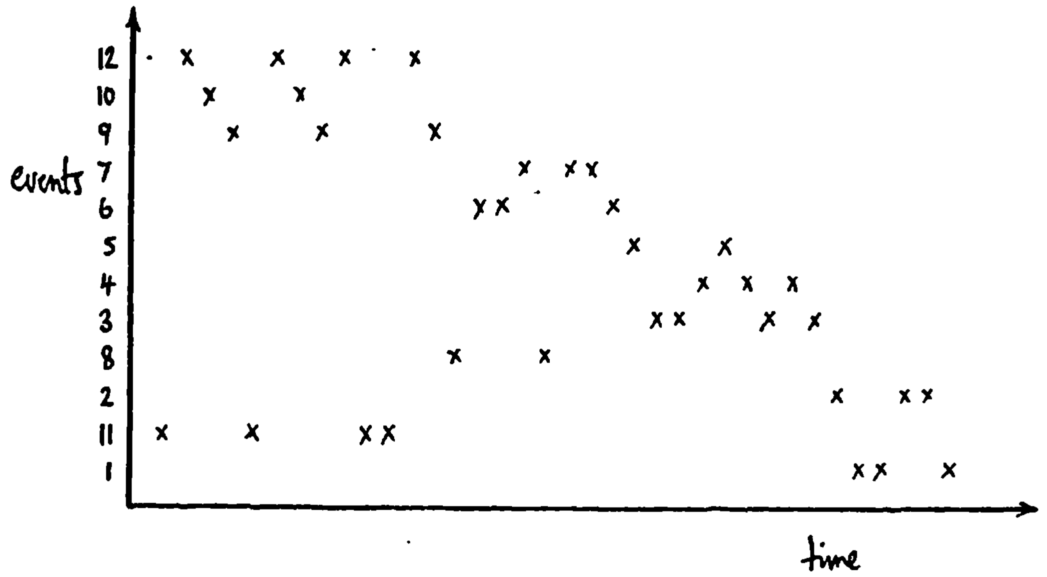


figure 60

The sequence of figure 59 with events re-arranged sophisticated control can be achieved with Markov Chain definition to vary the densities within the *tendency*.

If the probabilities in each row of the matrix are identical, i.e.

$$p_{ij} = \frac{1}{\nu_i}$$

then the density will be constant, but if the probabilities vary, differen effects will be obtained.

Example 6.4.4.1

Consider the simple expanding events range identified at the beginning of this section. If the matrix probabilities are weighted towards the diagonal, i.e.

$$\begin{array}{ll} p_{ij} < p_{i,j+1} & j = 1 , \ldots , i+1 \\ p_{ij} = 0 & j = i+2 , \ldots , n \end{array}$$

then the event sequence will be biased towards the 'new' events as represented in figure 61 below.

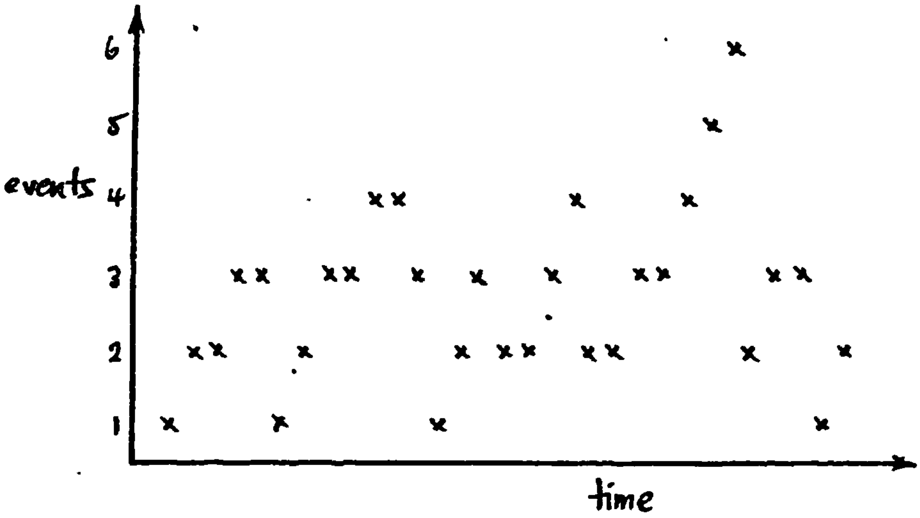


figure 61

An event sequence formed from a Markov Chain with a bias towards expansion

More subtle effects are possible by changing the probability weighting in each row. For example, the last row could have the bias reversed towards event e_1 :

$$p_{nj} > p_{n,j+1} \qquad j = 1 , \ldots , n-1$$

which will produce a series of 'escarpments' or a 'sawtooth' effect.

6.5 Extensions of Markov Chains

Before considering in detail a particular application of Markov Chains in computer music composition, brief mention will be made of some extensions of Markov Chains. No attempt is made to develop the mathematics of these systems, but there is scope for more research in each of these aspects. In some cases the mathematics appears to be quite complex, in others very little study, if any has been pursued.

6.5.1 Multi-dimensional Markov Chains

It is possible to extend the Markov Chains so far considered to a situation where the occurence probability of any event depends on the preceding two events:

$$p_{ijk} = p\left(E_{m+2} = e_k \mid E_{m+1} = e_j \wedge E_m = e_i\right)$$

In other words, p_{ijk} represents the probability of event e_j being followed by event e_k given that it was preceded by event e_i. These probabilities will be expressed in the form of a three-dimensional matrix.

Such a representation can be useful in applications involving analysis, where a more accurate characterization of sequence patterns in the original may be achieved than with a standard Markov Chain. However, if used to analyse and reconstruct only a short original sample event sequence, it is possible that the sequence will merely be reproduced wholesale.

When defining a matrix *ab initio* it is difficult to construct a three-dimensional Markov Chain to produce a required pattern. It is necessary to consider for each event, a set of ranges of following events associated with each possible preceding event. This may be conceptualized by associating one two-dimensional matrix M_j with each event e_j which sets out the probabilities p_{ik} that if event e_j is preceded by event e_i then it will be followed by event e_k. Such a task may be worthwhile if the number of events in the event space is small, but can otherwise become hopelessly complicated. The practical usefulness of three-dimensional Markov Chains is therefore questionable.

The author has in past work made use of three-dimensional Markov Chain structures with a randomly constructed probability matrix (Cf section 4.6) which is likely to produce clearly defined pattern structures. Simple three-dimensional Markov Chains defined over an event space of order two were used to generate the rhythms of *SKIRTRIKS*, the third movement of *MACRICISUM*. (See section 7.4).

The Markov Chain concept may be extended indefinitely to form multi-dimensional Markov Chains where the occurence probability of any event in a sequence depends on the preceding n events. But for the reasons cited above, the use of such structures is of questionable value.

6.5.2 Events defined as Markov Chains

The members of an event space may themselves be defined by Markov Chains. Two possible situations are identified:

1. Complex events in an event space may be defined at the start of the composing process using Markov Chains. Once the palette of events has been defined, they can be organized by the main executive Markov Chain according to the principles described above.

2. Events are not explicitly defined , but each is associated with a Markov Chain. Whenever the sequence building operation of the executive Markov Chain requires a particular event, then the Markov Chain associated with that event produces the required length of *sub-events*. Thus each event may be different in detail, but similar in character with each reappearance.

Considerable use of such 'chains of chains' has been made in earlier work of the author.[9]

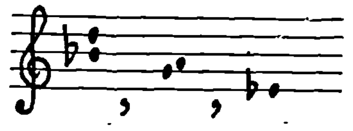
6.5.3 Markov Chains with event vectors

At the beginning of chapter four it was shown how individual events could be specified by the independent application of a number of event parameter spaces. If the number of parameters is N, then the character of any event can be represented by an *event vector* of order N . Markov Chains may then be used to produce sequences of event vectors, formed from the original event vector space.

The size of each event-parameter space need not necessarily be the same. In Example 4.2 the pitch space was of order 20, the duration space of order 16, the intensity space of order eight and the timbre space of order three. In general, a different Markov chain may be used to control each parameter, but if the order of each parameter space is the same, then the same Markov Chain may be used to control all of the parameters.

Example 6.5.3.1

Each event will be specified by an event vector of order three. The three parameter spaces also have order three and are defined as follows:

$E_{\text{pitch}} = \langle$  $\rangle$.

$E_{\text{rhythm}} = \langle$ ♪ , ♩ , ♩. $\rangle$

$E_{\text{intensity}} = \langle$ p , mf , ff $\rangle$

The same Markov Chain will be used to control each parameter. It is defined by the following matrix which has the event relation diagram of figure 62 (page 156).

$$P_{10} = \begin{bmatrix} \cdot 5 & \cdot 1 & \cdot 4 \\ \cdot 1 & \cdot 1 & \cdot 8 \\ \cdot 4 & \cdot 4 & \cdot 2 \end{bmatrix}$$

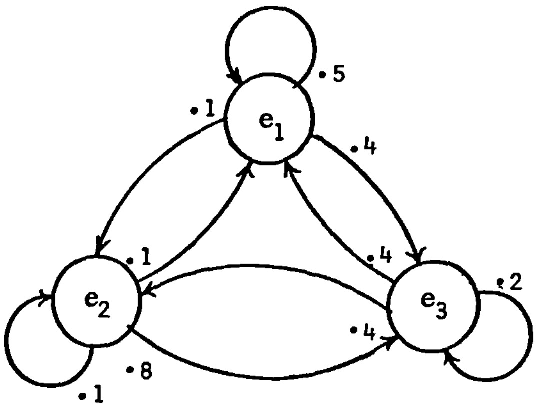


figure 62

A Markov Chain of order three

The following table shows two sequences of event vectors that might be produced:

1	1	1	1	1	1
1	1	1	1	3	1
1	1	1	3	1	1
3	1	3	3	1	3
1	1	1	3	3	1
1	3	1	3	1	1
1	1	3	3	1	3
1	3	3	2	1	1
1	3	1	2	3	3
2	1	1	2	3	2
3	3	1	3	1	3
3	2	1	2	1	1
2	3	3	3	2	3
3	2	2	2	3	2
3	2	2	3	1	3
1	3	1	1	1	2
1	2	3	1	1	2
3	3	3	1	1	3
			1	2	2

With the second sequence using pitches one octave lower, the result transcribes into figure 63 (page 157).

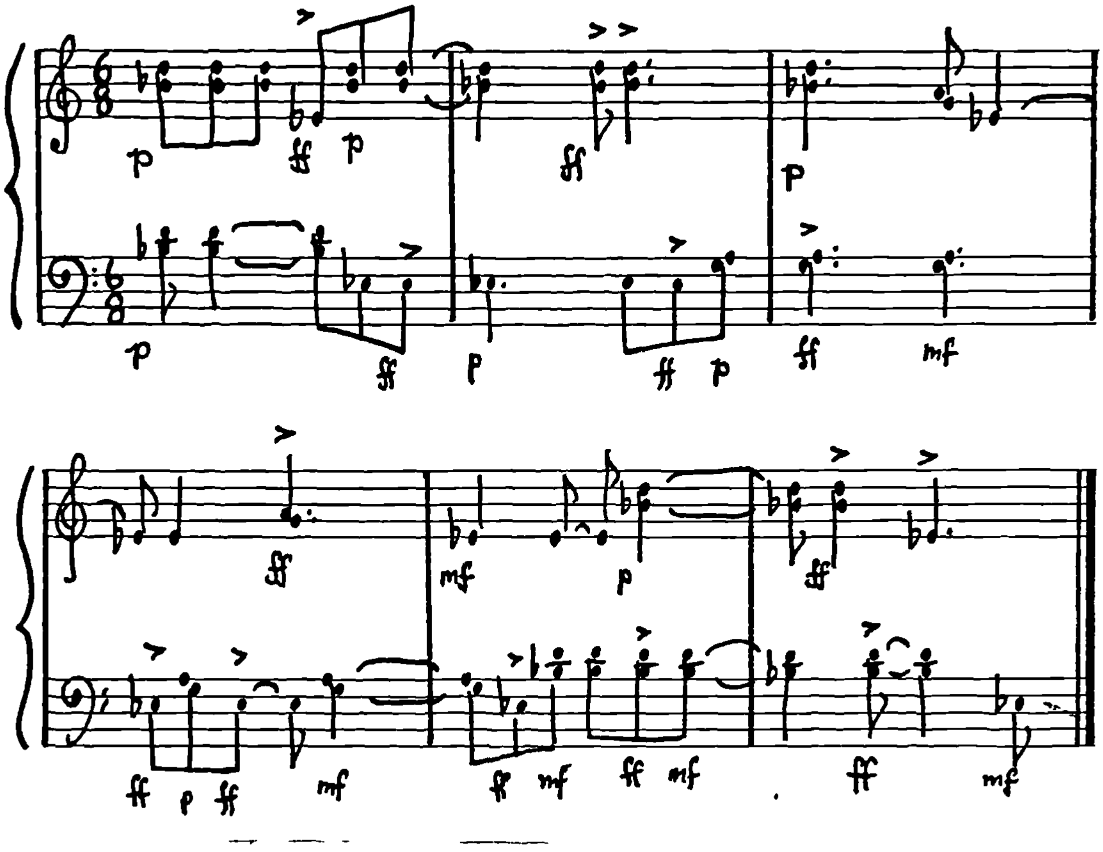


figure 63

A musical sequence generated by applying the Markov Chain of figure 62 over three event-parameter spaces

Markov Chains with event vectors were used throughout the composition *MACRICISUM*, the structure of which is considered in the following chapter.

6.5.4 Time variant probabilities

It is possible to define extended systems where the probabilities within the matrix change over a period of time. This may be done according to prescribed rules where it is possible to use a stochastic controlling system to determine the movement of probabilities. Alternatively, it is possible to adjust probabilities to respond to changes in the actual event environment being generated. This latter technique is a form of stochastic heuristic control where self-correcting or self-adapting possibilities may be built into a system.

The author has not yet pursued any research using or investigating structures of this type, but they have obvious potential.

CHAPTER SEVEN

APPLICATION IN COMPOSITION (II)

- MACRICISUM

The computer-generated tape composition *MACRICISUM* was developed and realised on the DEC System 10 computer at IRCAM in Paris during August 1977. Considerable use was made in the composition process of stochastic techniques, and forms of Markov Chains in particular, to generate and control the pattern structures of the work. The process of preparing data for digital sound generation was thereby greatly speeded up. From conception to listening to the completed sound realisation took a little under three weeks for a twenty minute composition. Such a work would have been quite impossible to produce without the use of especially derived and applied stochastic techniques which made the generation of unique and distinctive sounds possible. The overall plan of composition and the structuring of individual sections are described in the rest of this chapter.

7.1 Composing strategy

7.1.1 Semantic level

MACRICISUM was composed at the old IRCAM building in the Rue St. Merri, Beauborg in Paris. At the time of the composition workmen were busy outside completing the new IRCAM building, the hot August sun shone brightly from clear blue skies, and the windows were wide open. The accentuated aural environment was therefore full of the periodic sounds of hammering and drilling, punctuated and supported by the usual backgroun city noises of barking dogs, shouting tourists, accelerating motorbikes, car horns, church bells and the rumble of distant traffic. These effects, coupled with the startling architectural impact of the adjacent Beauborg Pompidou Centre in its bright cocooning matrix of steel pipes, glass, and plastic tubes, which nursed Xenakis' infant polytope under the venerable eye of the fifteenth century church of St. Merri across the square; all these contributed to influence the composer whilst working on the composition.

MACRICISUM is essentially a set of four textural studies, each based upon a particular generative scheme to produce a distinctive type of sound material. The four studies roughly emulate the form of a traditional symphony.

The first study, *SONATANOS*, is of a fairly cheerful nature and is in a quasi-sonata form. The second study, *LAITRAPARTIAL*, adopts a slower general pace and is more expansive. The third study, *SKIRTRIKS*, is a less serious scherzo-like movement. The last study, *DIRGEGRID*, is cast in the form of rondo-variations but with a mysterious, sometimes lugubrious and ultimately timeless mood.

7.1.2 Syntactic level

Making use of the stochastic structures described in the previous chapters, composing programs were written in SAIL (Stanford Artificial Intelligence Language), a language very similar to Algol but with input and output facilities adapted for specific use on the Stanford and IRCAM systems. The author has found that the sophisticated embedding facilities such a language offers are much more stimulating and attractive to work with than a more linear language such as FORTRAN which can be clumsy and restrictive to use.

The SAIL programs define general sets of constraints within which random choices are made to generate the data for sound synthesis. In general, control was exercised over *ranges of values* rather than *absolute values*. In most cases each possible range was split into two: a higher and lower range. The controlling Markov Chain was used to specify the order in which the ranges occured, then the actual parameter values were selected at random within that range.

The structure and application of the various programs are detailed in the following descriptions of the individual studies. The six SAIL generating programs which were written are summarized in the following list:

GRAND generates individual notes with straightforward random control of each parameter.

GSTOK generates blocks of notes with parameter ranges stochastically controlled by Markov Chains.

GRANP generates sound-masses with varying partial structure based on the priciples of Brownian motion

GRHY generates rhythms stochastically using three-dimensional Markov Chains.

GFM generates clusters of frequency-modulated pulsating pitches.

GGLIS generates multiple glissandi with the limits determined stochastically.

GRAND and GSTOK were used for the first study *SONATANOS*, GRANP and GRHY were used for the second and third studies respectively. The last study, *DIRGEGRID*, makes use of all six, but is mainly built up using GFM and GGLIS.

7.1.3 Sonic level: Mapping into a MUSIC 10 data file

The SAIL programs were designed to produce source-files for input to the digital sound synthesis system. The IRCAM MUSIC 10 language (often in the appropriate contexts referred to simply as MUSIC) for sound synthesis was used. This language has the same basic structure as Max Matthew's MUSIC 5 but with certain restrictions and extensions. It is designed specifically to be run on a DEC System 10 computer.

MUSIC 10 accepts lists of data defining a 'score' of specific note durations, intensities and whatever other parameters are being controlled. The input format is similar to the example discussed in section 3.1 and shown in figure 4 (page 38).

Each generating SAIL program, prefixed with the letter G, produced a MUSIC 10 source file identified by the same name as the SAIL program but without the prefix G. Thus:

GRAND	generated	a	MUSIC 10	source-file	called	RAND
GSTOK	"	"	"	"	"	STOK
GRANP	"	"	"	"	"	RANP
GRHY	"	"	"	"	"	RHY
GFM	"	"	"	"	"	FM
GGLIS	"	"	"	"	"	GLIS

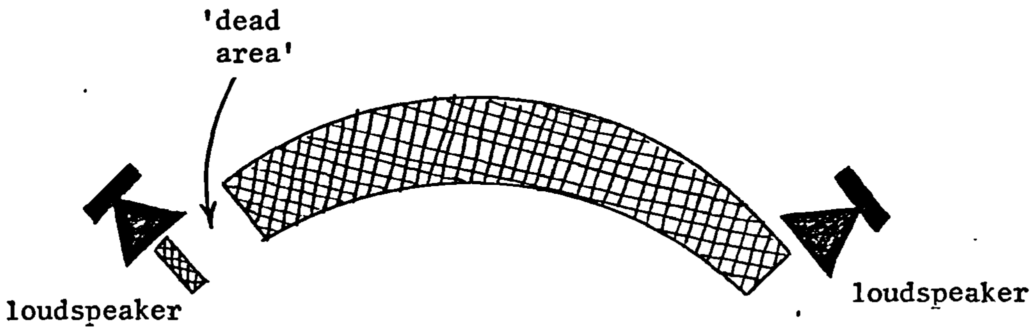
Because of limits in the amount of disc space available, work proceeded in short sections which were mixed using the digital mixing facilities available on the system. Working in stereo, with twelve-bit samples at a sampling rate of 12 800 per second, about ninety seconds was the maximum available length for individual sections. These could subsequently be spliced together digitally to build up each study. A few hundred runs were necessary to complete the composition, each taking between a few seconds and two hours to be completed.

Before proceeding to describe the studies in detail, two practical points need to be made, to account for some rather curious statements in the programs.

(1) SAIL uses its own unique input and output routines, undefined in standard ALGOL. In order to use the language to generate files, the format for the output statement required for the particular job was coaxed out of a resident programmer, and no other attempts were made to learn local dialectal variations. The programs were originally designed to be self-contained, and to be run without any external data. When it was later desired to vary the values of some of the constants in the program, it seemed quicker to change them directly, than to re-design the programs with special input statements. The programs compiled and ran almost immediately. The demands made on the system by a SAIL program were trivial compared to the vast processing capability subsequently required by a MUSIC 10 program for actual sound synthesis.

(2) An annoying disparity on using the system soon became apparent and necessitated some rather clumsy, but unavoidable additions to the generating programs. It was discovered that SAIL programs output numbers less than 0•1 in exponent notation using the symbol @ , which stands for "times ten to the power of", whereas the MUSIC 10 language accepts exponential notation with the symbol E , as in FORTRAN usage. Consequently MUSIC 10 rejected @ as an unidentifiable symbol. In the absence of any systems programmers the problem was solved in the following way. To avoid the generation of any @'s by SAIL programs a test was made to check if any generated values lay between 0 and 0•1. If so, they were

rounded up or down to exactly 0 or 0·1 . In general this did not affect the final sound output in any perceivable way. The most significant effect concerns the spatial positioning of sounds across the stereophonic spectrum. The position of a sound is specified in MUSIC 10 by a value between 0 at the extreme left and 1 at the extreme right. The inability to generate a value between 0 and 0·1 means that there is a gap for the first tenth of the space between left and right speakers (see figure 64).



<u>figure 64</u>

<u>*Sound discontinuity in the stereo space found in parts of MACRICISUM*</u>

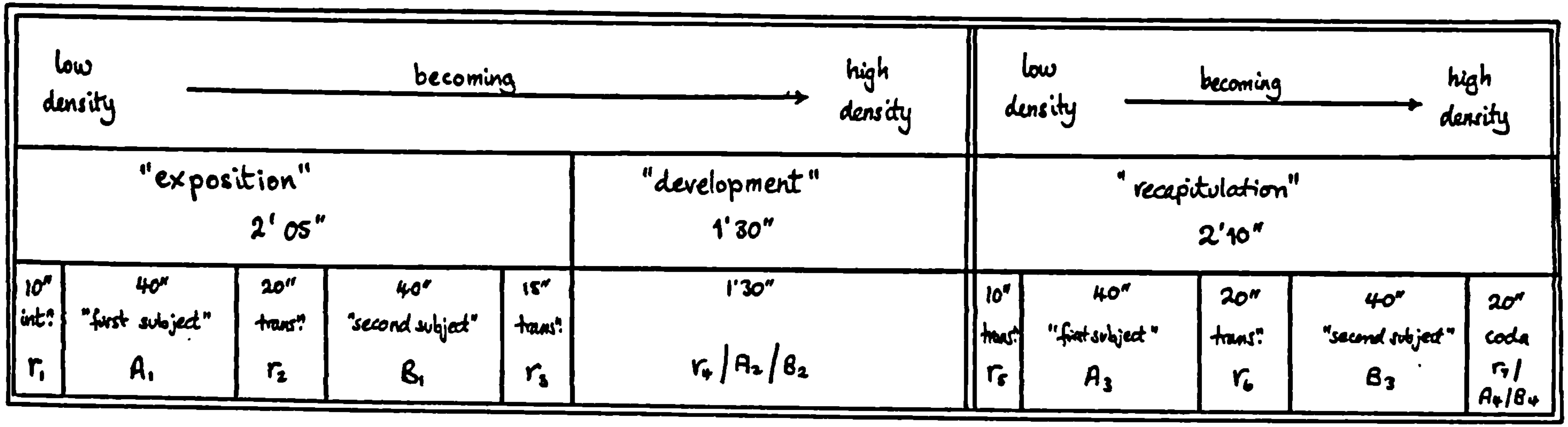
This means that whenever a sound travels across from one side to the other, on reaching the gap it will hover for a moment before jumping to the other side. Fortunately such a subtle effect is virtually undetectable by the ear. Occasionally it was necessary to specify an amount less than 0·1 , for example in the second study *LAITRAPARTIAL*, where a sound is redefined every one twentieth of a second. In this case the value 0·05 is specified as a constant amount, and was written on to the MUSIC 10 source-file as such. It is only when a *variable* amount between 0 and 0·1 is required that problems arise.

7.2 Study One: *SONATANOS*

7.2.1 Sectional Organization

SONATANOS was organized on a structural plan similar to classical sonata form (hence the title). The main 'subjects' were generated using Markov Chains with event vectors to define the parameters governing the placing and nature of blocks of sound. Different 'subjects' were generated by changing the probabilities of the Markov Chains. Where further subject matter was generated from the same set of Markov Chains it was possible to produce consistent material essentially the same in character but with varying details. The introduction, transitions and coda were generated using sounds whose parameter values were selected totally at random. The 'development' section was a free mixture of both types of generation.

On to the basic sonata form shape was superimposed an overall increase in density of events moving to a climax at the end of the development section, then falling to build up again to the end of the study. The overall structural design is represented by figure 65 (page 167). The two generative structures and their corresponding programs used for the study are described in the following sections.



<table>
<tr><td colspan="6">low density —— becoming ——→ high density</td><td colspan="5">low density —— becoming ——→ high density</td></tr>
<tr><td colspan="5">"exposition"
2' 05"</td><td>"development"
1'30"</td><td colspan="5">"recapitulation"
2'10"</td></tr>
<tr><td>10"
int:
r_1</td><td>40"<br>"first subject"
A_1</td><td>20"
trans:
r_2</td><td>40"<br>"second subject"
B_1</td><td>15"
trans:
r_3</td><td>1'30"
$r_4 / A_2 / B_2$</td><td>10"
trans:
r_5</td><td>40"<br>"first subject"
A_3</td><td>20"
trans:
r_6</td><td>40"<br>"second subject"
B_3</td><td>20"
coda
$r_7 /$
A_4 / B_4</td></tr>
</table>

figure 65

The overall structural design of SONATANOS

7.2.2 Transitional Sections: The GRAND program

For the transitional sections, introduction and coda, a simple generating program GRAND was written which gave random control over the duration, pitch, intensity, envelope shape, harmonic content and spatial position of each sound in a sequence. In the program all the parameter values are chosen from within a specified range. In the case of *envelope shape* and *harmonic content* two functions are specified for each from which a simple choice may be made.

The GRAND program to generate 'random' sounds is listed in appendix D1 (page 317). The program is reproduced below with a commentary on each section.

```
BEGIN
   INTEGER I;
   REAL S,T,R;
   LABEL SKIP;
   BOOLEAN FLAG;
   OPEN(1,"DSK",Ø,Ø,2,Ø,Ø,Ø);
```

The variables to be used in the program are declared and output channel 1 is prepared for writing to disk.

```
ENTER(1,"RAND",FLAG);
```

The file RAN to which output is to be written is identified for output channel 1.

```
CPRINT(1,
   "INSTRUMENT TOOT;
   OSCIL(P4,MAG/P2,P5);
   OSCIL(U1,MAG*P3,P6);
   OUTA ← OUTA+U2*P7;
   OUTB ← OUTB+U2*(1-P7);
   END;
```

The instrument definition is written into the output file RAND. This simple instrument, called TOOT, has variable duration (P2), pitch (P3), amplitude (P4), envelope (P5), harmonic spectrum (P6) and spatial position (P7) control. Pi refers to the i-th item in each note definition in the note list which is to follow.

```
NCHNS ← 2;
ARRAY F1,F2,F3,F4(512);
```

The number of output channels is set to two, and array space is declared for four functions.

```
SEG(F1); Ø,1  1,2Ø  .5,6Ø  Ø,1ØØ;
SEG(F2); Ø,1  1,15  .5,3Ø  .5,75  Ø,1ØØ;
```

Two envelope functions F1 and F2 specifying varying attack and decay times, are defined using the line segment generator SEG. They have the shapes plotted in figure 66 (page 170).

```
SYNTH(F3); 1,1  2,.9  3,.8  4,.4  6,.3  8,.2  999;
SYNTH(F4); 1,1  3,.7  5,.5  7,.5  9,.5  11,.3  999;
```

Two harmonic functions F3 and F4 are defined using the SYNTH facility which adds together harmonics in the relative strengths specified to produce their resultant waveform. The functions have the spectra plotted in figure 67 (page 170).

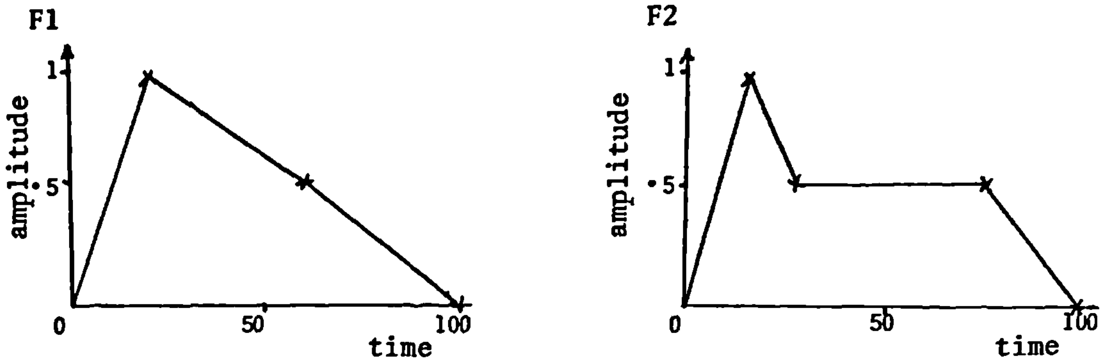


figure 66

Two envelope functions used in SONATANOS

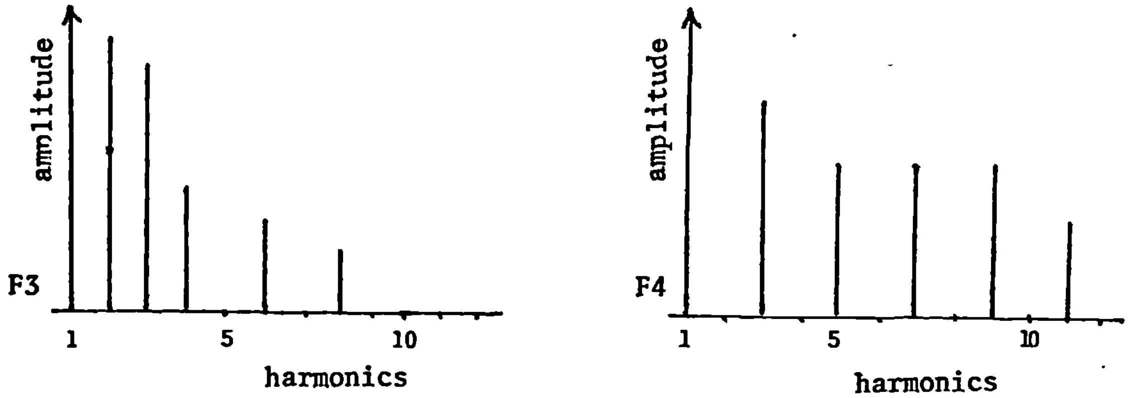


figure 67

Two sets of harmonic spectra used in SONATANOS

```
PLAY;
");
```

The instruction to play the following note list is added to the output file RAND, which concludes the first PRINT instruction to the file. Having specified the fixed material at the start of the output file RAND, the rest of the program calculates the variable note list which forms the main body of the file.

```
FOR I←1 STEP 1 UNTIL 4Ø DO S←RAN(Ø);
```

The SAIL function RAN(Ø) will deliver a random number in the range 0 to 1. However, the random sequence will always begin with the same number and produce an identical sequence each time the program is run. Thus the above dummy statement was introduced to generate a number of spare values before the main part of the program began, with the number of generations, 40 in this particular listing, being changed each time the program was used to produce a different result.

```
S←Ø;
```

The starting time is initialized to zero.

```
FOR I←1 STEP 1 UNTIL 6Ø DO
BEGIN "LOOP"
```

The following loop is computed for each line of note data required, up to a maximum of 60 notes.

```
T←RAN(Ø)*2; R←RAN(Ø);
```

T, the length of the note, is computed. It may assume any value between 0 and 2 seconds. The value of R, which indicates the spatial position, is computed.

```
IF T<Ø.1 THEN T←Ø.1;
IF R<Ø.1 THEN R←Ø;
```

T and R are checked to see if their value is less than 0·1 . If so, an adjustment is made for the reason explained above (page 164/5).

```
CPRINT(1,"TOOT",S,T,RAN(Ø)*RAN(Ø)*7ØØØ,
       RAN(Ø)*1ØØ+3ØØ,
```

The start of a line of note data is printed to the output file. "TOOT" specifies the 'instrument' on which the note is to be played (there is only one in this program), followed by S the starting time, and T the duration. The next expression evaluates the pitch, between 0 and 7 000 Hertz. The double use of the random function builds in a bias towards lower values to compensate for the logarithmic pitch scale. The next expression evaluates the intensity, expressed by a value between 300 and 400 .

```
IF RAN(Ø) > Ø.5 THEN " F1 " ELSE " F2 ",
IF RAN(Ø) > Ø.5 THEN " F3 " ELSE " F4 ",
```

An even choice is made between the two envelope functions and the two harmonic spectrum functions.

```
R,";
");
```

The spatial position value R is printed. The line of data is concluded by printing a semicolon and a move to a new line is specified. The current print instruction is brought to an end.

```
S ← S+T+RAN(Ø);
```

The start time S is incremented by the value of the last note duration, and up to one second of silence is added.

```
IF S > 2Ø THEN GOTO SKIP;
```

This instruction causes an escape from the loop if more than 20 seconds of sound has been defined.

```
    END "LOOP";
```

Assuming a skip out has not occured, the loop is repeated for up to 60 lines of note data.

```
    SKIP: CPRINT(1,"FINISH;");
```

With the note list complete, the MUSIC 10 data is concluded with the "FINISH" statement.

```
    CLOSE(1);
    RELEASE(1);
  END
```

The output file is closed and the channel released to complete the program.

A typical listing of a line of note data might have the following form:

```
  TOOT  4.5  1.7  578  245  F1  F4  .35;
```

It can be seen that the above program GRAND is essentially writing another program RAND in the MUSIC 10 language.

Having considered this program in detail, it will not be necessary to cover the remaining programs in the same depth, but only to identify the significant instructions peculiar to each.

The generative program GRAND and its offspring RAND were run a number of times to generate the material for each transitional section of the piece, and the results were mixed digitally to build up the required sound.

Where the density is low, as at the beginning, only two runs were required. For a higher density up to six or seven mixes were used.

Making use of such a strictly controlled random definition of parameter values produces a result which sounds far from chaotic. This is evident at the very beginning of *SONATANOS*. Of course, if the material was allowed to continue indefinitely it would lack any clear overall shape or direction, but in short sections interesting and clear sound patterns may be discerned.

7.2.3 Main subjects: the GSTOK program

The main subjects are built up from sequences of comolex events consisting of *sound blocks*, the limits of which are defined by an *event vector* of parameter values. Each event vector is of order nine. The nine parameters concerned in defining the sound block limits are:

1. Pitch range
2. Overall pitch area
3. Block length
4. Density
5. Intensity
6. Following silence
7. Envelope of each element
8. Harmonic spectrum of each element
9. Spatial position

Each of these parameters can assume just two values 1 or 2 to specify whether each falls in the higher or lower range, or a choice between two alternatives. Having specified the ranges, the individual elements which go towards making up a sound block are chosen at random within the specified limits.

Example 7.2.3.1

The event vector (2,2,1,2,1,1,1,2,1) will define a short, dense, loud block coming from the left-hand channel. It will cover a wide pitch range, being composed of higher pitches using the first of two specified envelopes and the second of two specified harmonic spectra. There will only be a short gap between it and the following block.

The inverse of such a block would be long, of low density, soft and coming from the right-hand channel. It would cover a small pitch range being composed of low pitches using the second of two specified envelopes and the first of two specified harmonic spectra. There would be a long gap before the next block. The corresponding event vector is (1,1,2,1,2,2,2,1,2).

The sequence of parameter range values in the event vectors are controlled by a set of nine corresponding two-dimensional Markov Chains. For any two-dimensional Markov Chain it is only necessary to store the values of p_{11} and p_{21} since $p_{12} = 1-p_{11}$ and $p_{22} = 1-p_{12}$. Thus the matrices defining the nine Markov Chains can be stored in a 9×2 array.

Example 7.2.3.2

Suppose the matrix M_3, controlling the third parameter-range, had the following form:

$$M_3 = \begin{bmatrix} \cdot 8 & \cdot 2 \\ \cdot 5 & \cdot 5 \end{bmatrix}$$

It would mean that short block lengths will tend to be followed by more short block lengths, but long block lengths may be followed equally by either long or short. Notice that the stationary distribution of such a matrix will be biased towards event parameter 1 , for short block lengths will in general be more likely to occur.

Example 7.2.3.3

Suppose each of the nine matrices has the following form:

$$M_i = \begin{bmatrix} \cdot 8 & \cdot 2 \\ \cdot 2 & \cdot 8 \end{bmatrix} \qquad i = 1 , \ldots , 9$$

This means that any parameter-range value will tend to stay the same, four times out of five. The stationary distribution will be (·5 , ·5) for in general, both parameters are equally likely to occur. The sequence of event vectors likely to be produced by such a set of matrices may be like the following:

$$\begin{array}{lcl}
E_1 & = & \langle\ 2\ \ 1\ \ 2\ \ 2\ \ 1\ \ 2\ \ 1\ \ 2\ \ 1\ \rangle \\
E_2 & = & \langle\ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2\ \ 1\ \rangle \\
E_3 & = & \langle\ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 1\ \ 2\ \ 2\ \rangle \\
E_4 & = & \langle\ 1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 2\ \ 1\ \ 2\ \rangle \\
E_5 & = & \langle\ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \rangle \\
E_6 & = & \langle\ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \rangle \\
E_7 & = & \langle\ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \rangle \\
E_8 & = & \langle\ 2\ \ 2\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1\ \ 2\ \ 2\ \rangle \\
E_9 & = & \langle\ 2\ \ 2\ \ 1\ \ 1\ \ 2\ \ 1\ \ 1\ \ 2\ \ 2\ \rangle \\
E_{10} & = & \langle\ 2\ \ 2\ \ 1\ \ 2\ \ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \rangle
\end{array}$$

(The pattern produced by such matrices has much in common with one of the structuring devices used in Koenig's Project One program.)

The structure described by this sequence can be represented diagrammatically by figure 68 (page 178/9).

In the program itself, all the matrices are constructed with probabilities determined randomly. It is therefore highly unlikely that the matrices will be the same. Some parameters may tend to alternate in range, with others one range may be totally dominant.

The program which stochastically arranges sound blocks in this way, GSTOK, may be found in Appendix D2 (page 318). In the program simple functions are necessary to change the parameter range values 1 and 2 to actual parameter limits within which the sound elements are defined.

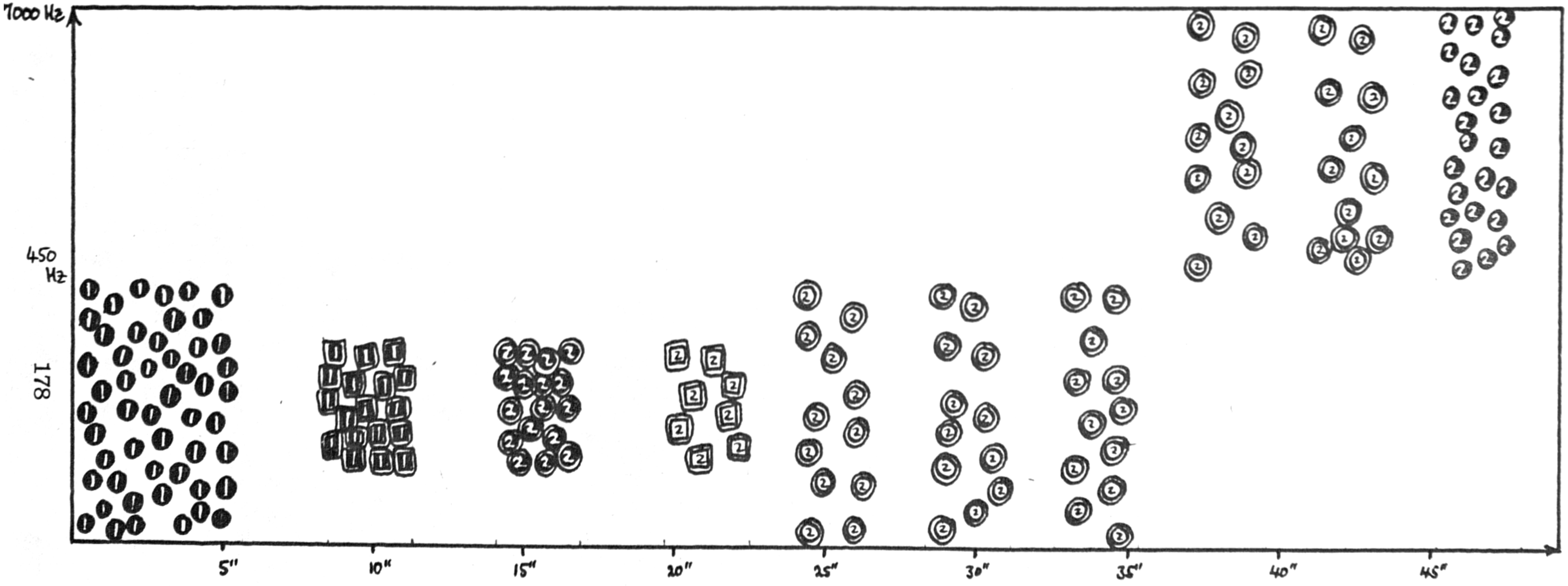


figure 68

Representation of a structure formed by applying the same two-event Markov Chain to nine different event-parameter spaces

Key to figure 68

The pitch range, pitch area, block length, density and following silence parameters are all clear. The other parameters are represented by the following symbols:

Parameter 5: intensity: no boundary (e.g. ○ or ■), low - 1
with boundary (e.g. ◉ or ⧈), high - 2

Parameter 7: envelope: circle (e.g. ◉ or ○), type 1
square (e.g. □ or ▣), type 2

Parameter 8: harmonic: black (e.g. ⓵ or ❷), type 1
white (e.g. ② or 1⃞), type 2

Parameter 9: spatial position: = the number inside the circle or square. 1 is left, 2 is right

The GSTOK program

The initial statements are similar to those used in GRAND, The MUSIC 10 instrument "TOOT" is the same as was used before with identical envelopes and harmonic spectra.

```
FOR I←1 STEP 1 UNTIL 7 DO J←RAN(Ø);
```

The number of iterations of this dummy loop are changed for each run requiring a different set of stochastic matrices.

```
FOR I←1 STEP 1 UNTIL 9 DO
FOR J←1 STEP 1 UNTIL 2 DO
P [I,J]←RAN(Ø);
```

The probabilities in the stochastic matrices are determined at random.

```
FOR J←1 STEP 1 UNTIL 3 DO J←RAN(Ø);
```

The number of iterations of this dummy loop are changed for each run requiring a different sound sequence generated from the <u>same</u> set of stochastic matrices.

```
FOR I←1 STEP 1 UNTIL 9 DO
V[I]←RAN(Ø)+1;
```

The nine initial parameter range values of the event vector V[ ] are determined at random. V[ ] is defined as an integer array, and so the result of evaluating RAN(Ø)+1 will be automatically rounded up or down to 1 or 2 for storing in the array.

```
FOR I←Ø.1 STEP 1 UNTIL 7 DO
BEGIN "I"
```

The loop to generate a sequence of sound blocks is begun. I is the starting time of each block and the loop is repeated until the required number of seconds, seven in this case, has been reached.

```
FOR J←1 STEP 1 UNTIL 9 DO
IF RND(Ø)<P[J,V[J]]
THEN V[J]←1 ELSE V[J]←2;
```

The new values of the event vector V[ ] are determined from the stochastic matrices.

```
FOR J←1 STEP 1 UNTIL (V[3]+1)*15 DO
BEGIN "J"
```

The "J" loop calculates the elements in each sound block and writes the necessary list to the data file. Computation is in units one fifteenth of a second long, and so the loop will generate two or three seconds of sound depending upon the block length specified in the third entry of the event vector.

```
J←J+RAN(Ø)*V[4];
```

The space between elements is varied by incrementinp J by up to 2 units according to the density specified in V[4] .

```
CPRINT(1,"TOOT",I+J/15,
      (IF RAN(Ø)<Ø.5 THEN "Ø.Ø6 " ELSE " Ø.Ø5 "),
```

The line of note data specifying each element is headed by the instrument name "TOOT", followed by the starting time, derived from I and J . For the next value in the list, the note duration, a simple random choice is made between 0·05 and 0·06 . This means that the elements will not be repeated with a mechanical regularity, but the timing will be slightly variable. This will not affect the overall density or length specifications.

```
    (375*V[2]-35∅)*(3-V[1])*(2↑(V[1]+1)*RAN(∅)+1),
```

The pitch is chosen to fall within the boundaries specified by the pitch area and pitch range parameters, with a bias towards lower frequencies to compensate for the logarithmic pitch scale.

If V[1] and V[2] both equal 1, then the pitch will be between 50 and 250 Hz. If V[1] = 1 and V[2] = 2, the pitch will be between 800 and 4 000 Hz. If V[1] = 2 and V[2] = 1, it will be between 50 and 450 Hz, and if both V[1] and V[2] equal 2 then the pitch will be between 400 and 3 600 Hz.

```
    1∅∅∅*V[5]-5∅∅,
    (IF V[7]=1 THEN " F1 " ELSE " F2 "),
    (IF V[8]=1 THEN " F3 " ELSE " F4 "),
```

The intensity is set at 1 500 or 500. The envelope and harmonic spectrum functions are selected.

```
    V[9]-1,";
    ");
  END "J";
```

The spatial position is set at 0 or 1 and then the line of note data is completed with a semicolon and a move to a new line. The "J" loop is then repeated for the required number of times.

```
    IF V[6]=2 THEN I←I+1;
    I←I+V[3]+1;
    END "I";
```

I is incremented by the length of the completed sound block, and an extra second is added depending upon the specification for following silence in parameter V[6]. The "I" loop is then repeated until the required sequence length has been produced. The closing lines of the program are the same as for GRAND.

For each section requiring material generated from GSTOK a number of runs were made and the results mixed digitally. The precise number of layers used depended upon the overall density specifications of the study, as with the GRAND generated transitional sections.

The aural effect of using these structuring techniques is to produce grouped clusters of jerky, pointed pitches. The resulting sounds hover on the perceptual boundaries between continuous mingled textures and discreetly identifiable notes. The durations were chosen to deliberately exploit such a potential imbroglio. This can create a certain degree of tension as the ear attempts to identify note sequences and fails; it cannot cope with such an unfamiliar texture. Halving the playback speed of the study doubles the lengths and produces quite a different impression. Some listeners have expressed a preference for the slower version where the sound material falls into a more familiar and reassuring category.

Most of the time it is rather difficult to hear the transitional relationships between consecutive sound block parameter specifications, but the general stationary distribution of parameter values is more obvious. There are sections of the study where a definite dominance of sound from one channel may be discerned, or where more frequent periods of silence are apparent, for example.

7.3 Study Two: *LAITRAPARTIAL*

7.3.1 Sound-mass specification (Micro-structuring)

LAITRAPARTIAL is built up with sound masses moulded from a conceptual *fundamental* on which is built a continuously varying partial structure. Twenty possible partials are redefined every twentieth of a second. Thus each micro-event lasting for 0•05 seconds, is specified by two linked vectors each of order twenty: a *partial vector* specifying which harmonics are present in the sound at that instant, and an *amplitude* vector specifying the relative strength of each harmonic. The movement of the vector quantities is based on the principle of Brownian motion, or the "Random Walk" (see section 6.4.1 page 130). Thus each value specifying an harmonic in the partial vector will be incremented up or down by one to define the next micro-event. Similarly each element in the amplitude vector will be incremented or decremented by a small amount. The fundamental itself is not necessarily present.

Thus if the following represents the start of a possible partial vector definition:

3 5 9 13 16 . . . .

when redefined for the next micro-event it may become:

2 6 10 12 17 . . . .

and so on.

Similarly, the corresponding amplitude vector which represents the relative strengths of each specified harmonic, may go through the following sequence:

$$\begin{array}{cccccccccccc}
(\cdot 3 & \cdot 4 & \cdot 9 & \cdot 3 & \cdot 4 & . & . & . & . &) \\
(\cdot 2 & \cdot 5 & 1 & \cdot 4 & \cdot 5 & . & . & . & . &) \\
(\cdot 3 & \cdot 4 & \cdot 9 & \cdot 5 & \cdot 4 & . & . & . & . &) \\
(\cdot 4 & \cdot 3 & \cdot 8 & \cdot 4 & \cdot 3 & . & . & . & . &)
\end{array}$$

In general the random walk has reflecting barriers which allow a sequence to change direction when a limit is reached. The exception is with the higher harmonics which are not given an upper limit.

The movement of particles of sound from harmonic to harmonic, and the changes in their relative amplitude cause different particle strands to move in and out of prominence. The effect may be represented diagrammatically by figure 69(a) (page 186).

Additional constraints in the definition of each sound mass or macro-event, provide for a possible bias towards upward or downward movement. If the bias is towards upward movement, the sound particles will sweep through the upper harmonics and disappear twinkling out of the audible range (figure 69(b)). If the bias is towards downward movement, the particles will all pile down together to settle on to the pulsating fundamental (figure 69(c)).

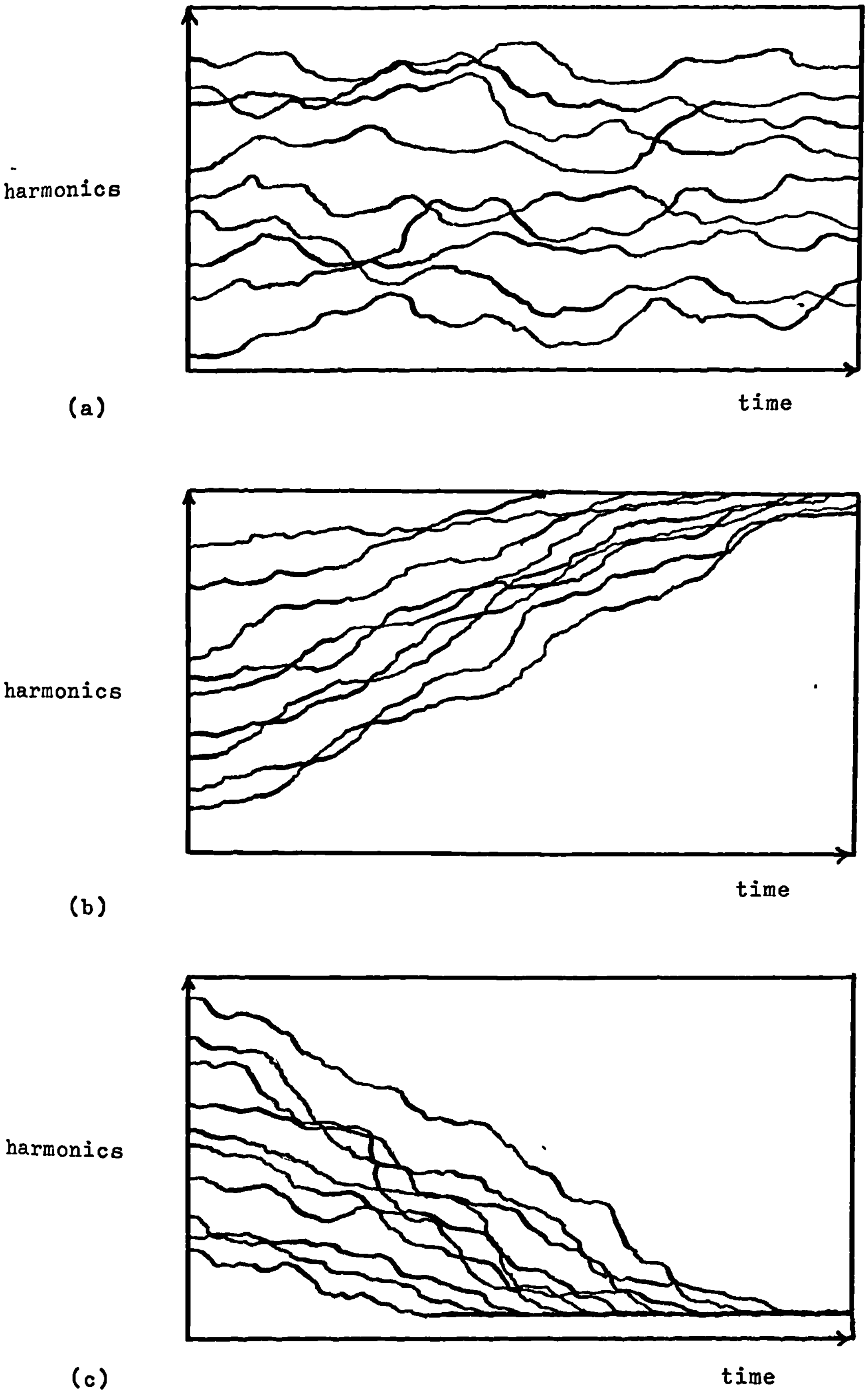


figure 69

Representation of the movement through adjacent partials of a number of strands of sound

7.3.2 Dynamic macro-structuring

In the first study the nine parameters used were randomly defined as fixed values for a particular sound block. In *LAITRAPARTIAL* use is made of *dynamic macro-structuring* where the parameter values can change according to a specified pattern. So, for example, instead of the spatial parameter values 1 and 2 specifying that a sound should come consistently from either the left or the right, they will specify *movement* from right to left, or from left to right. In this study there are six parameters involved for macro-event control. They are:

V[1] : length of macro-event, variable within limits
V[2] : intensity change; soft to loud or loud to soft
V[3] : spatial change; left to right or right to left
V[4] : pitch change bias, whether present or not
V[5] : only used if V[4] = 2, specifies whether the bias is up or down
V[6] : length of gap between macro-events, variable between limits

With the parameter value defined as above, the structure is built up in the same way as the sound blocks were arranged in *SONATANOS* with a set of Markov Chains controlling the sequence of vectors.

The study was completed in two 90 second long sections, each generated from its own set of Markov Chains. For each half, the program was run three times to create three layers which were digitally mixed together. With the harmonic structure being redefined every twentieth of a second, up to two thousand harmonic changes occur. For this reason the sound generating MUSIC 10 program, RANP took up to one and a half hours to run, which is very long by IRCAM standards. By comparison, most of the other programs used in *MACRICISUM* took just a few minutes to synthesize the required sound. A small amount of pseudo-reverberation was added by repeatedly mixing the resulting digital sound file with itself, delayed slightly and at a lower amplitude.

7.3.3 The GRANP program

The introductory material and instrument definition are the same as before.

```
SEG(F2); 1,1  1,1ØØ;
```

The *envelope* for each micro-event is a simple straight line function, always at maximum (figure 70).

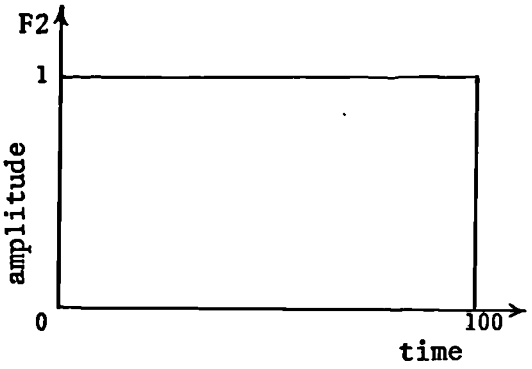


figure 70

The micro-event envelope function used in LAITRAPARTIAL

The actual macro-event *envelope* is created as a result of the intensity changes specified by V[2] .

```
FOR I←Ø.1 STEP 1 UNTIL 9Ø DO
BEGIN "I"
```

Ninety seconds of sound will be created by the "I" loop.

```
PITCH←RAN(Ø)*RAN(Ø)*4ØØ+3Ø;
```

The fundamental is chosen between 30 and 400 Hz with a bias towards lower pitches. It is kept reasonably low since high harmonics are to be built upon it, but it does not go lower than 30 to avoid the generation of clicks, should all the sound particles end up on the fundamental.

```
PARTIAL[1]←15*RAN(Ø);
AMP[1]←1;
```

The values of the first entry in the starting partial and amplitude vectors are chosen.

```
FOR K←2 STEP 1 UNTIL 2Ø DO
BEGIN
   PARTIAL[K]←PARTIAL{K-1]+1Ø*RAN(Ø);
   AMP[K]←RAN(Ø);
   IF AMP[K]<Ø.1 THEN AMP[K]←Ø.1;
END;
```

The remaining nineteen entries of the two vectors are computed.

```
FOR J←1 STEP 1 UNTIL 6 DO
IF RAN(Ø)<P[J,V[J]] THEN V[J]←1 ELSE V[J]←2;
```

The new values of the macro-event vector V[ ] are computed.

```
LENGTH←18Ø*(RAN(Ø)+V[1] - 1);
```

The length of the macro-event, in twentieths of a second, is computed. It will be between 0 and 9 or 9 and 18 seconds depending upon the value of V[1] .

```
FOR J←LENGTH/1Ø STEP 1 UNTIL LENGTH*Ø.9 DO
BEGIN "J"
```

The "J" loop calculates the micro-events to complete the required length of macro-event. The curious start and finish limits are to avoid the generation of numbers less than 0•1 later in the program.

```
CPRINT(1,"SYNTH(F3);");
FOR K←1 STEP 1 UNTIL 2Ø DO
CPRINT(1,PARTIAL[K],",",AMP[K]," ");
CPRINT(1,"999;
");
```

The line of MUSIC 10 data to define the harmonics present in the micro-event is written to the output file RANP by listing each entry of the PARTIAL vector and the AMP vector in turn, which defines the function F3.

```
CPRINT(1,"STER ",I+J/2Ø." Ø.Ø5 ",PITCH,
```

The line of note data is begun. It is headed by the instrument name, "STER" in this case. The starting time of the note is derived from I and J. It is followed by the duration, always 0•05, and the fundamental pitch which was derived above and will remain the same throughout the "J" loop.

```
12ØØ/LENGTH*(IF V[2]=1 THEN J ELSE LENGTH-J),
" F2 F3 ",
```

The intensity is derived from the length and value of J. As the "J" loop progresses it will either move from a maximum of 1 200 down to 1 to create an overall diminuendo across the macro-event, or from 1 to 1 200, depending upon the value of V[2]. The basic harmonic and envelope function specifications F2 and F3 are constant for each note though, of course, the actual definition of F3 is changed before each note definition by the preceding "SYNTH" statement.

```
    (IF V[3]=1 THEN J/LENGTH ELSE 1-J/LENGTH),";
    ");
```

The spatial position is similarly related to the J and LENGTH values, being incremented for each note to create a gradual movement of sound from one side to the other across the macro-event, the direction depending upon V[3]. The line of note data is terminated with a semi-colon, and a new line is prepared.

```
    FOR K←1 STEP 1 UNTIL 2Ø DO
    BEGIN "K"
```

The "K" loop (which is still inside the "J" loop) calculates the new partial and amplitude vectors.

```
    PARTIAL[K]←PARTIAL[K]+
    (IF V[4]=1 THEN 2*RAN(Ø)-1
               ELSE IF V[5]=1 THEN RAN(Ø)
                              ELSE - RAN(Ø));
    IF PARTIAL[K] < 1 THEN PARTIAL[K]←1;
```

Each old value in the PARTIAL vector is incremented up or down by one, with the possibility of a bias towards upward or downward movement on the settings of V[4] and V[5]. A test is made to see if the lower limit has been overstepped, in which case the value is brought back within the limit.

```
    AMP[K]←AMP[K]+ (IF RAN(Ø) > Ø.5 THEN Ø.1 ELSE -Ø.1);
    IF AMP[K] < Ø.1 THEN AMP[K]←Ø.1;
    IF AMP[K] > 1   THEN AMP[K]←1;
  END "K";
```

Each element in the AMP vector is incremented up or down by 0·1 . Checks on upper and lower limits are made, which completes the "K" loop.

```
        END "J";
        I ← I+LENGTH/2Ø+7.5*(RAN(Ø)+V[6]-1);
    END "I";
```

The "J" loop is continued for the required length of macro-event. When that is completed, the value of I (in seconds) is incremented by the macro-event length (scaled from twentieths of a second) and up to 15 seconds of silence is added depending upon the V[6] specification. The "I" loop is then repeated to create more macro-events until the required total length of ninety seconds has been reached.

The aural effect of the structuring methods used in *LAITRAPARTIAL* is to produce crystalline shimmering sheets of sound full of complex intermingling harmonics; reminiscent of the sort of resonances that might be produced by the free vibration of dangling groups of vast metal panels.

It was expected that the results should have a realistic sound, having been built up to exploit natural harmonic resonances, with rapid incremental changes in structure emulating the dynamic variation of natural oscillations. Nevertheless, the richness and energy of the resulting sound textures came as a very pleasant surprise when the output was first heard.

7.4 Study Three: *SKIRTRIKS*

7.4.1 Rhythmic organization and macro-structure

SKIRTRIKS (the title is derived from the words *Scherzo* and its English meaning *Tricks*) is a study in rhythmic organization where a number of interlocking rhythmic strings are defined. Once a random pitch has been assigned to a string it remains constant. As with *LAITRAPARTIAL* above, a set of 2×2 Markov Chains is used to control the major parameter limits for defining each string. The parameters identified by the parameter vector V[ ] are the following:

V[1] : length of string, long or short
V[2] : direction, whether static or changing
V[3] : if V[2] = 1 then position of sound, left or right
V[4] : if V[2] = 2 then whether movement is towards the left or towards the right
V[5] : intensity, whether static or changing
V[6] : if V[5] = 1 then whether soft or loud
V[7] : if V[5] = 2 then whether diminuendo or crescendo
V[8] : general pitch of string, high or low
V[9] : length of following silence, long or short
V[10] : general speed of the rhythmic string, fast or slow

The program was run eight times with the same set of matrices to form an eight-layered texture. Two sections, each using a different set of matrices, were completed to make up the study.

7.4.2 The use of three-dimensional stochastic matrices

To generate the rhythmic strings themselves, three-dimensional (2 × 2 × 2) stochastic matrices were used where the occurence probability of an event depends upon the preceding two events. In this way a much tighter and more regulated structure is built up than is the case with standard two-dimensional matrices. Here, when applied to generate rhythms, it also means that a structure with a greater metrical unity is obtained. A simple two event space was used with one note twice as long as the other. These may for the sake of convenience be represented as a crotchet and a quaver. So for example, if figure 71 below represents the most likely event to follow each possible pair of events, then a rhythmic structure will result where crotchet and quaver tend to alternate, but whenever an event does succeed in following the same event, that pattern is likely to persist, as demonstrated in the sequence of figure 72.

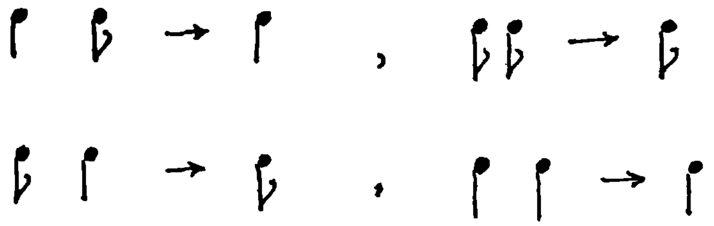


figure 71

Rhythmic pairs and the associated rhythmic value by which each is most likely to be followed

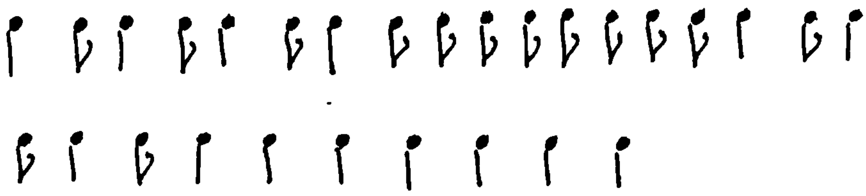


figure 72

Rhythmic sequence formed using the eventualities of figure 71

If, on the other hand, the inverse situation applies, as represented by figure 73, then a pattern sequence such as that in figure 74 may ensue.

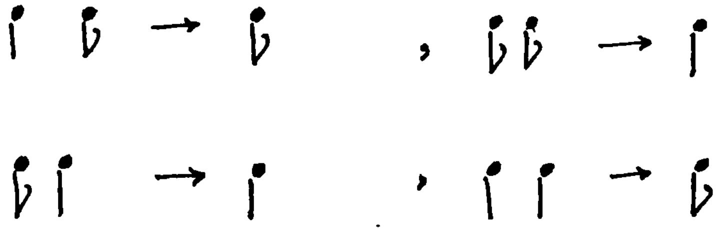


figure 73

Inverse eventualities to figure 71

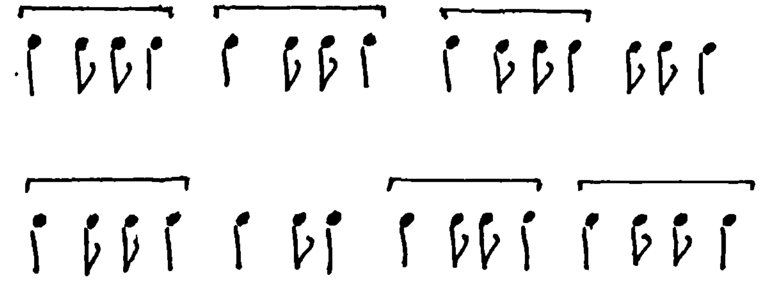
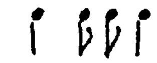


figure 74

A rhythmic sequence formed using the eventualities of figure 73

This pattern has a basic metrical 'feel' of $\frac{3}{4}$ about it, but the use of stochastic procedures means that the basic  rhythm occasionally slips, momentarily destroying the pulse, and then picks up again. This type of result is characteristic of the study as a whole and gives it a certain human-like quality. It sounds very much as if perhaps a group of children were playing together, but not quite in time, and making occasional mistakes.

7.4.3 Sonic Mapping and effects

The instrument defined in MUSIC 10 on which the notes were to be played consisted of filtered noise of very narrow bandwidth with each note having a sharp attack and a steep decay. This produced percussive sounds with a quality similar to that of a marimba or xylophone, and even sometimes sounding like a piano. This enhanced the effect already mentioned above in producing natural sounding results, very similar in character to the sound of an Indonesian Gamelan Orchestra. In fact, after hearing the tape, certain listeners have thought that the sounds had been recorded by live performers playing on actual instruments, possibly following a computer-composed score; whereas, of course, all of the sounds are produced by direct digital synthesis.

7.4.4 The GRHY program

The basic structure of the program, which may be found in full in appendix D4, is similar to that of the programs already described above. Those lines of the program which are particularly different or significant are listed below followed by an explanation or commentary.

```
FOR I←1,2 DO FOR J←1,2 DO
RP[I,J]←RAN(Ø);
```

The rhythm probabilities are stored in the matrix RP[ ]. In the same way as a two-dimensional, two-event stochastic matrix requires only a one-dimensional array for storage, as mentioned earlier, so a three-dimensional, two-event stochastic matrix requires only a two-dimensional matrix for storage. The above statement constructs the required matrix, with random values.

```
RH1←RAN(Ø)+1; RH2←RAN(Ø)+1;
```

The *current* rhythmic event value is stored in RH2. RH1 holds the *last* rhythmic event value. The above statement gives these two variables starting values.

```
SPEED←Ø.Ø75*(RAN(Ø)+V[1Ø]-1)+Ø.1;
```

The speed of the rhythmic string, which is the length of the basic time unit, is set between 0•1 and 0•175 seconds if V[10] = 1 and between 0•175 and 0•25 seconds if V[10] = 2 .

```
LENGTH←1Ø*(RAN(Ø)+V[2]-1)/SPEED;
```

The length of the string is set between 0 and 10 seconds if V[2] = 1 and between 10 and 20 seconds if V[2] = 2 . This value is then divided by SPEED to give the number of actual notes in the string, stored in the variable LENGTH.

```
VOL←5ØØ + 1ØØØ*(V[6]-1);
POS←Ø.45*(RAN(Ø)+V[3]-1)+Ø.1;
```

The volume is set at 500 or 1 500 depending on the value of V[6]. The position is set between 0•1 and 0•55 if V[3] = 1 and 0•55 and 1•0 if V[3] = 2 .

```
FOR J←1 STEP 1 UNTIL LENGTH DO
BEGIN "J"
```

The "J" loop, to determine and print the specification for each note in the string, is begun.

```
RH3←RH2;
```

RH3 is a temporary variable used to hold the old value of RH2 while RH2 assumes a new value.

```
RH2←(IF RAN(Ø)<RP[RH1,RH2] THEN 1 ELSE 2);
```

The new value of RH2 is determined from the matrix RP[ ], using its old value, and also the value of the previous event RH1.

```
RH1←RH3;
```

RH1 assumes RH2's old value which had temporarily been stored in RH3.

```
      CPRINT(1,"NOIS ");
```

The heading to the line of note data is written to ouput channel 1.

```
      CPRINT(1,I+J*SPEED,SPEED,PITCH," ",
```

The line of note data is begun. The number of the note, J, is multiplied by the note speed to give the time into the string in seconds, and to this is added the current value of I, the master loop within which a series of rhythmic strings are being formed. The note duration (=SPEED) and PITCH are written to the ouput file. These remain constant for the whole string.

```
      (IF V[5]=1 THEN VOL
                  ELSE 15ØØ/LENGTH*(IF V[6]=1 THEN J ELSE LENGTH-J));
```

The volume is set to the constant value VOL, worked out above, if V[5] = 1. Otherwise the volume is gradually incremented for the duration of the string, up or down depending on the value of V[6].

```
      (IF V[2]=1 THEN POS
       ELSE (IF V[4]=1 THEN J ELSE LENGTH-J)/LENGTH*Ø.9+Ø.1,";
```

Similarly the position is either set to the constant value POS or incremented up or down depending on the value of V[2] and V[4].

```
      J ← J+RH2;
```

Since a 'wooden' percussive instrument is being simulated, all the notes will have the same length. So the actual effect of a longer note will be to delay the start of the following note. This instruction accordingly adds the value of RH2 to J to fix the starting time of the following note and thus generate the required rhythmic pattern.

7.5 Study Four: *DIRGEGRID*

7.5.1 Sectional organization

The final study, *DIRGEGRID*, is built up from two types of contrasting material which are arranged alternately in a sequence similar to a rondo/ variation form. The two types of material consist of groups of pulsating clusters of frequency modulated notes, and dense grids of multiple glissandi. A short central 'development' section includes further material generated from the same programs as were used in the preceding studies, and a final coda completes the study. The basic structure is represented diagrammatically by figure 75.

7.5.2 Frequency modulated pulsating sound clusters

A simple FM 'instrument' was used, which produces time-variant timbres. In addition, forty simultaneous layers of note sequences were generated, with varying durations and intensities. As with *SKIRTRIKS* above, groups of note strands were generated with each strand having its own consistent pitch. But in this case, the note durations were totally dependent on the pitch. In addition, no use was made of Markov Chain control over the major parameters, which were simply selcted at random. Since the generated sections are rather short, as with the GRAND sections of the first study *SONATANOS*, and since the parameters are not independent, the result sounds far from random.

"EXPOSITION/VARIATION"					"DEVELOPMENT"	"RECAPITULATION/EXTENSION"				"CODA"
15"	30"	15"	30"	15"	30"	15"	30"	15"	30"	30"
FM$_1$	Glissandi$_1$ 80 voices even distribution	FM$_2$	Glissandi$_2$ 80 voices downward bias	FM$_3$	material generated from GRAND, GSTOK, GRANP and GRHY	FM$_4$	Glissandi$_3$ 160 voices even distribution	FM$_5$	Glissandi$_4$ 80 voices upward bias	FM$_6$ FM cluster repeated 8 times slow diminuendo

figure 75

The overall structural design of DIRGEGRID

The relationship between pitch and note length can be represented diagrammatically by figure 76. Higher pitches have short durations, lower pitches have long durations.

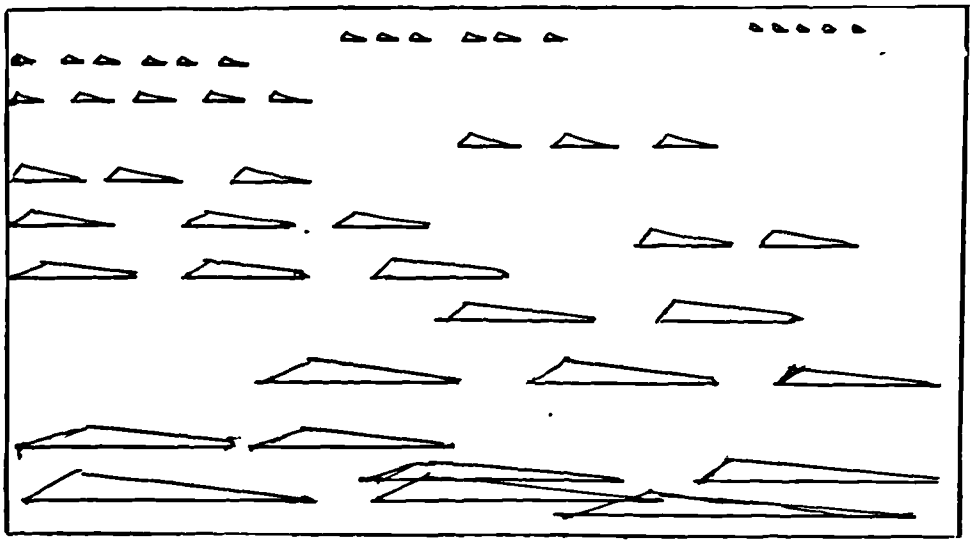


figure 76

Representation of the result of linking pitch and note lengths in the FM sections of DIRGEGRID

The GFM program

The complete program may be found in appendix F5.

```
FOR K←1 STEP 1 UNTIL 4Ø DO
FOR I←L STEP 1 UNTIL TOTLEN DO
BEGIN "I"
```

The "K" loop generates 40 layers. The "I" loop generates each layer.

```
PITCH←RAN(Ø)*RAN(Ø)*6ØØØ+5Ø;
LENGTH←1Ø-PITCH↑Ø.5/8;
```

The pitch, constant for each string, is chosen between 50 and 6 050 Hz with a bias towards lower values to compensate for the logarithmic pitch scale.

The note length is calculated from the pitch. This will vary from 0·3 seconds, if the pitch is at 6 050 Hz to just over 9 seconds if the pitch is at 50 Hz, again, with an adjustment for the logarithmic pitch scale. Figure 77 shows the curve which relates pitch and note length.

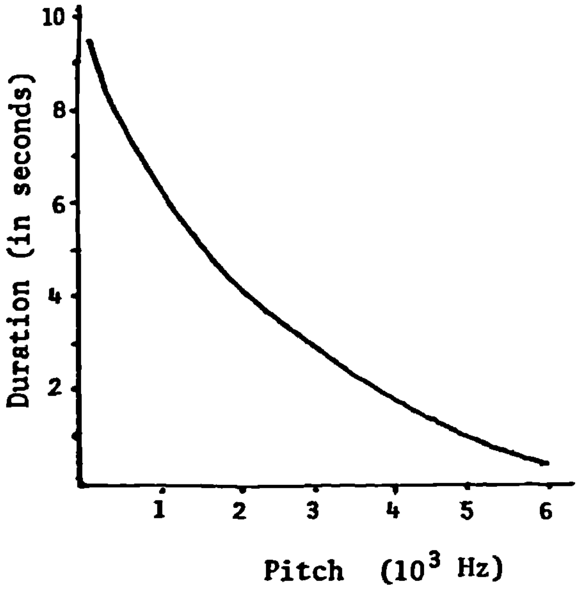


figure 77

The function relating pitch and note duration in the FM sections of DIRGEGRID

```
FOR J←1 STEP 1 UNTIL RAN(Ø)*1Ø DO
BEGIN "J"
```

The "J" loop is repeated for the number of notes in the string, between 0 and 9 .

```
      CPRINT("
      FM",K," ",I," ",LENGTH," ",
```

The line of note data is begun. It is headed by the instrument name: FM, followed by K: the instrument number, then the starting time: I, and the note length.

```
      PITCH," ",RAN(Ø)*1ØØ+1ØØ," F1 F3 F2 ",
```

The pitch is added, followed by the intensity which can vary freely between 100 and 200 Hz, and then function designators for envelope and frequency modulation control.

```
      RAN(Ø)*Ø.9+Ø.1,";");
```

The line of note data is concluded with the spatial specification, between 0·1 and 1·0 .

```
      I ← I+LENGTH+RAN(Ø)*LENGTH*(TOTLEN-I)/2;
      IF I > TOTLEN THEN GOTO STOP;
   END "J"; STOP:
   END "I";
```

I is incremented by the note length and an additional amount which varies with the value of LENGTH and which gets shorter as the end of the layer is approached. If the required length has been reached, the process stops. Otherwise, the "J" loop is repeated for the required string length, then a different pitch is chosen, and the process repeated for another string.

The patterning structure means that each FM block begins with a cluster of notes starting simultaneously, the components of which then proceed to vary in different, but related ways. The fast repetition of the higher pitches and the slow repetition of the low pitches produces a strange pulling or dragging effect, whilst the random changes in relative intensity and spatial characteristics produce some odd results: the sounds sometimes appear to 'bend' in space or pitch. This gives the study as a whole a mysterious, distant and alien quality.

7.5.3 Multiple glissandi

A set of Markov Chains was again used to control the major parameters specifying how, where and when the glissandi occur. The eight values in the parameter value array correspond to the following parameters:

V[1]	:	length of glissando, long or short
V[2]	:	root pitch of glissando, high or low
V[3]	:	envelope of glissando, slow decay (= diminuendo) or slow attack (= crescendo)
V[4]	:	harmonic content of glissando 'note'
V[5]	:	range of glissando, wide or narrow
V[6]	:	direction of glissando, up or down
V[7]	:	spatial location, from left or right
V[8]	:	length of following silence, long or short

Four separate glissandi sections were generated, one section in 160 voices and the other three in 80 voices. Of these three, one had a bias to move to lower pitches towards the end of the section, one had a bias to move to higher pitches towards the end of the section, and the other was without bias.

The GGLIS program

The complete program may be found in appendix D6.

```
SYNTH(F3);1,1  3,.6  5,.4  7,.3  9,.25  11,.2  13,.1  999;
SYNTH(F4);1,1  2,.6  4,.4  6,.3  8,.25  1Ø,.2  12,.1  999;
```

Two harmonic functions F3 and F4 are defined. F3 is built on odd-numbered harmonics, F4 on even. These have the spectra shown in figure 78.

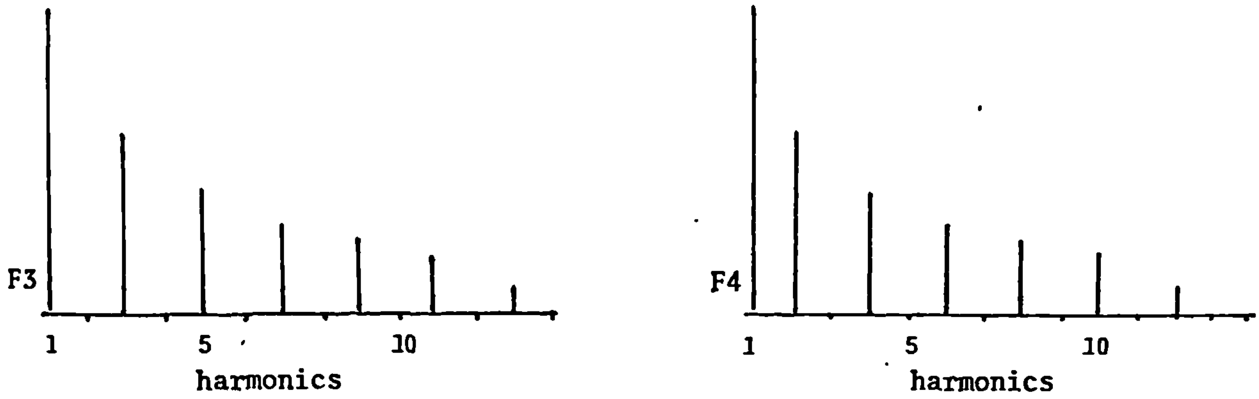


figure 78

The harmonic spectra used in the GLIS sections of DIRGEGRID

```
SEG(F5);Ø,1   .8,7   1,12   Ø,1ØØ;
SEG(F6);Ø,Ø   .3,5Ø  1,1ØØ;
SEG(F7);1,Ø   .3,5Ø  Ø,1ØØ;
SEG(F8);Ø,1   1,95   Ø,1ØØ;
```

Two envelope functions are defined, F5 and F8, which are plotted in figure 79. The two functions F6 and F7 are used for determining the slope of the glissando. These simple pseudo-exponential curves are used to compensate for the logarithmic pitch scale. They ensure that a glissando does not suddenly appear to swoop very swiftly into the lower frequencies, or alternatively appear to slow down on approaching the higher frequencies, but instead maintains a reasonably even gradient. F7 and F6 are plotted in figure 80.

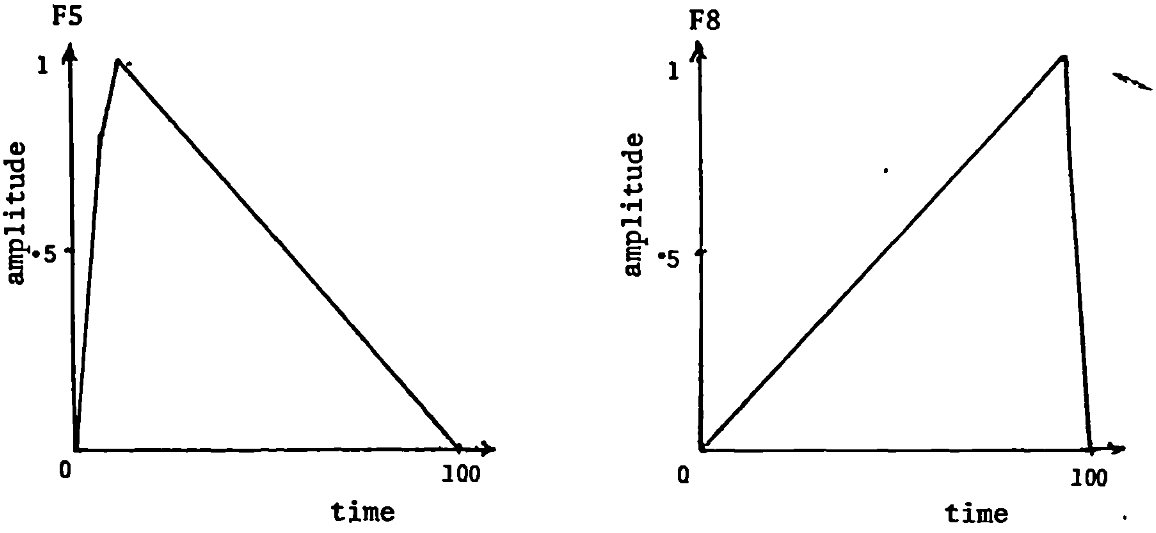


figure 79

The envelope functions used in the GLIS sections of DIRGEGRID

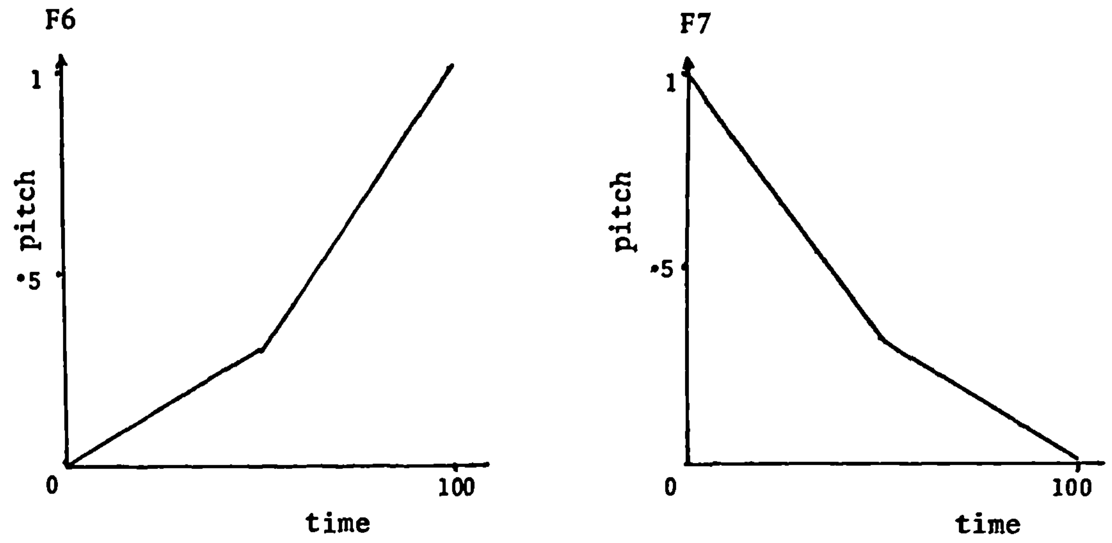


figure 80

The slope functions used in the GLIS sections of DIRGEGRID

```
FOR K←1 STEP 1 UNTIL 8Ø DO
FOR I←1 STEP 1 UNTIL 2Ø DO
BEGIN
```

The "K" loop is repeated 80 times to generate 80 layers of glissandi. The "I" loop is repeated for each layer, to generate at least 20 seconds of sound.

```
FOR J←1 STEP 1 UNTIL 8 DO
V[J]←(IF RAN(Ø)<P[J,V[J]] THEN 1 ELSE 2);
```

Before each glissando is defined, the "J" loop generates fresh values for the parameter vector V[ ] from the set of stochastic matrices stored in the P[ ] matrix.

```
CPRINT(1,"
        GL ",K," ",I," ",LENGTH←5*(RAN(Ø)+V[1]-1)," ",
```

The line of note data is begun with the instrument name "GL", and number "K", followed by the starting time "I" and the note length. LENGTH may vary between 0 and 5 seconds if V[1] = 1 and 5 and 10 seconds if V[1] = 2 . If the I loop is begun with I = 20, and the length is chosen to be 10 seconds, then the length of the whole block will be 30 seconds, which is the maximum possible. This accounts for the program statement that the "I" loop should be repeated for up to 20 seconds in order to generate the 30 second long sections identified in the study plan of figure 75 (page 202).

```
PITCH←(21-I)*1Ø*(2↑V[2]*RAN(Ø)*RAN(Ø)+V[2])+25,
```

The root pitch is calculated. At the beginning of the block (I=1), it will be between 225 and 625 Hz if V[2] = 1 and between 425 and 1 225 if V[2] = 2 . At the end of the block (I=20) it will be between 35 and 55 if V[2] = 1 and between 45 and 85 if V[2] = 2 . Thus the average pitch will drop throughout the loop. The expression (21-I) was replaced simply with I , to accomplish the reverse: a rising of average pitch through the loop. To maintain a constant average pitch throughout, the expression (21-I)*1Ø was replaced with the value 5Ø. In evaluating PITCH, compensation is once again made for the logarithmic pitch scale.

```
" 2ØØ ", IF V[3]=1 THEN " F5 " ELSE " F8 ",
IF V[4]=1 THEN " F3 " ELSE " F4 ",
```

The average intensity is set consistently at 200. The envelope is chosen according to the value of V[3] , and likewise the harmonic spectrum according to V[4] .

```
PITCH+PITCH*1.5*(RAN(Ø)+V[5]-1),
```

The pitch range of the glissando is evaluated, dependent upon the PITCH. If PITCH is at its minimum, 35, then the pitch range will lie between 35 and 87 if V[5] = 1 and between 87 and 140 if V[5] = 2. If PITCH is at its maximum, 1 225, then the pitch range will lie between 1 225 and 3113 if V[5] = 1 and between 3113 and 4900 if V[5] = 2. It should be pointed out that the value PITCH identifies the root, or lowest pitch of the glissando from which it will rise, or on to which it will descend.

```
IF V[6]=1 THEN " F6 " ELSE " F7 ",
```

The slope function to control the glissando is chosen; up or down depending upon the value of V[6].

```
.45*(RAN(Ø)+V[7]-1)+Ø.1,";");
```

The spatial location is chosen to lie between 0·1 and 0·55 if V[7] = 1 and between 0·55 and 1·0 if V[7] = 2.

```
I ← I+LENGTH+(IF V[8]=1 THEN Ø ELSE 5);
```

I is incremented by the length of the glissando and an additional amount of 1 or 6 seconds depending upon the value of V[8].

The dense and aggressive, yet sinister thrust of the glissandi sections contrasts strongly with the more translucent sound of the interspersed FM sections. The heavy density creates a thick and sometimes stodgy texture which comes through quite forcefully even when at a low volume. But inside the heavy texture vibrant movement can be discerned. At the less dense edges of the sections, stray glissandi appear to escape and mingle with the FM sounds, so that the blocks merge into each other.

7.6 Performance and Assessment

In any concert performance of musical tape compositions particular problems of presentation arise. Apart from a few enthusiasts who are happy to sit and critically assess the standard of a professional quality play-back system as a contribution to the live concert experience, the vast majority of the public find that the lack of any visual component in a public presentation, to which they are accustomed in a normal concert, forms a considerable barrier. Although some elitists would maintain that there is nothing better than the uncluttered dissemination of pure sound, there is no doubt that simple tape playback concerts are not popular with most concert goers. It is more suitable to listen to pure tape music in a domestic environment, where perhaps headphones may be used. Indeed, some more popular types of electronic music, produced entirely in the studio with electronic means, have achieved considerable commercial success without ever having been performed in public.

It seemed appropriate that since *MACRICISUM* was deliberately constructed to create distinctive types of sound textures, that analogies with similar patterns in the natural visual environment should be exploited. Accordingly, for public presentation of *MACRICISUM* a sequence of slides has been prepared which explore the correlation between the sound and visual textures. Examples of the types of illustrations used may be found in figures 81 to 84.

In the first study, *SONATANOS*, the single, randomly specified notes were associated with images containing simple, spaced components, such as the small group of wasps in figure 81(a). These contrasted or extended into

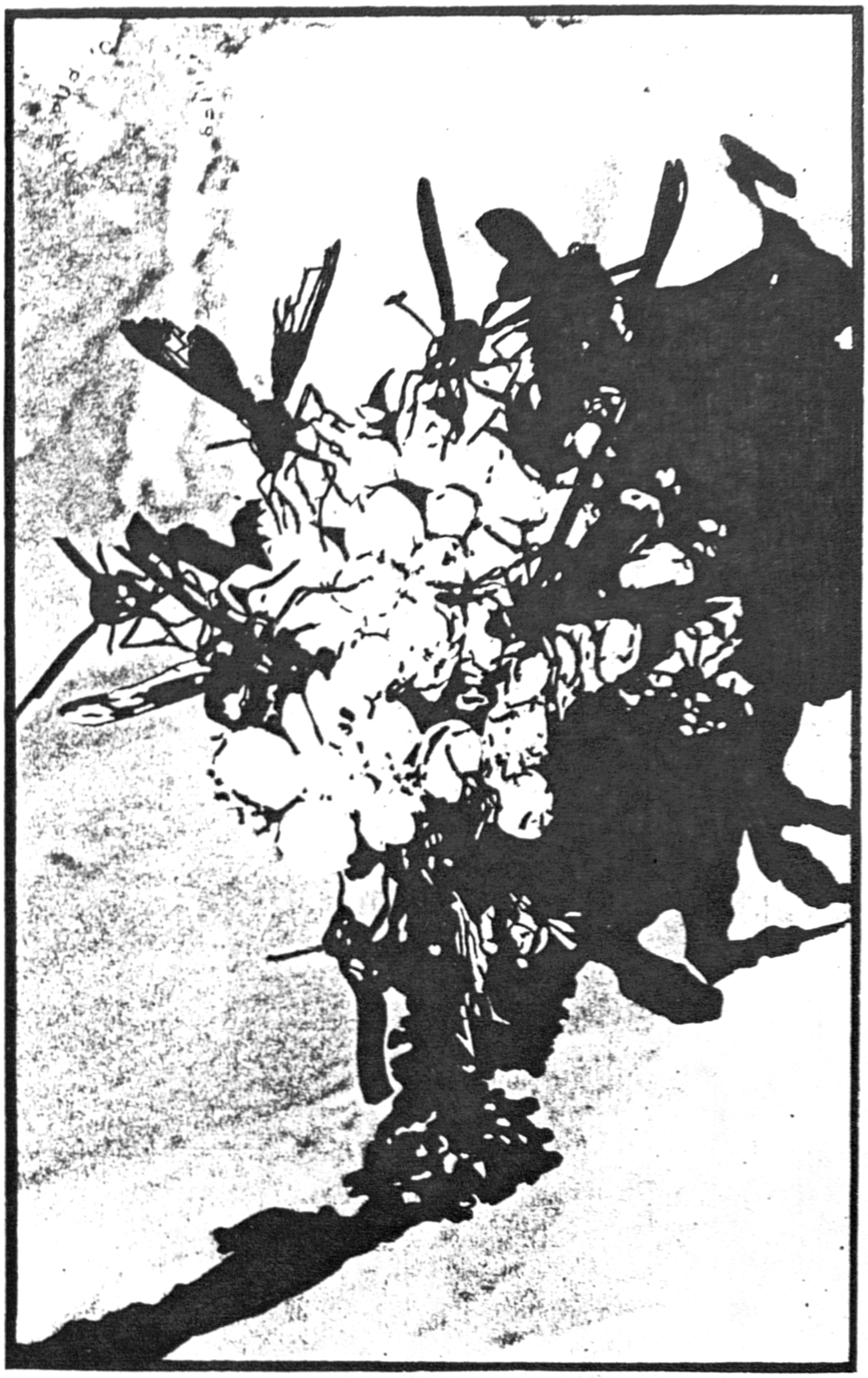


figure 81(a)

Objects which tend to gather in clusters around a core; for example, insect colonies.

groups of clustered objects corresponding to the stochastically structured sound blocks. The related images here are represented by a flock of migrating birds (figure 81(b)) and a group of ants clustering around their queen (figure 81(c)). The actual jerky and spluttering sounds of this study bear a strong resemblence to those made by groups of insects, and further emphasize the visual connection.

The second study, *LAITRAPARTIAL*, uses structures where large numbers of small components are constrained to remain within a particular path or shape. Here a relation has been identified with a colony of marching ants (figure 82(b)) and a galaxy of stars (figure 82(a)).

The third study, *SKIRTRIKS*, makes use of overlapping groups of periodically structured objects. The piping on the outside of a chemical refinery (figure 83(a)) and in the engine room of a large ocean-going supertanker (figure 83(b)) are ideal visual equivalents. They have a marked resemblance to the exterior of the Centre Pompidou, Beauborg, under whose shadow the composition of *MACRICISUM* was undertaken.

The glissandi of the fourth study *DIRGEGRID* have an obvious correlation with criss-crossed scaffolding networks (figure 84(a)) and spiders' webs (figure 84(c)), whilst the mysterious FM clusters and pulses are reflected in the music of the spheres: strange distant galaxies and nebulae in space (figure 84(b)). Figure 84(a), being an old teleseope mounting, unifies these two ideas. The image of the spider's family and web in figure 84(c) additionally draws together threads from the whole group of studies, elements of which recur at the centre of this final study.

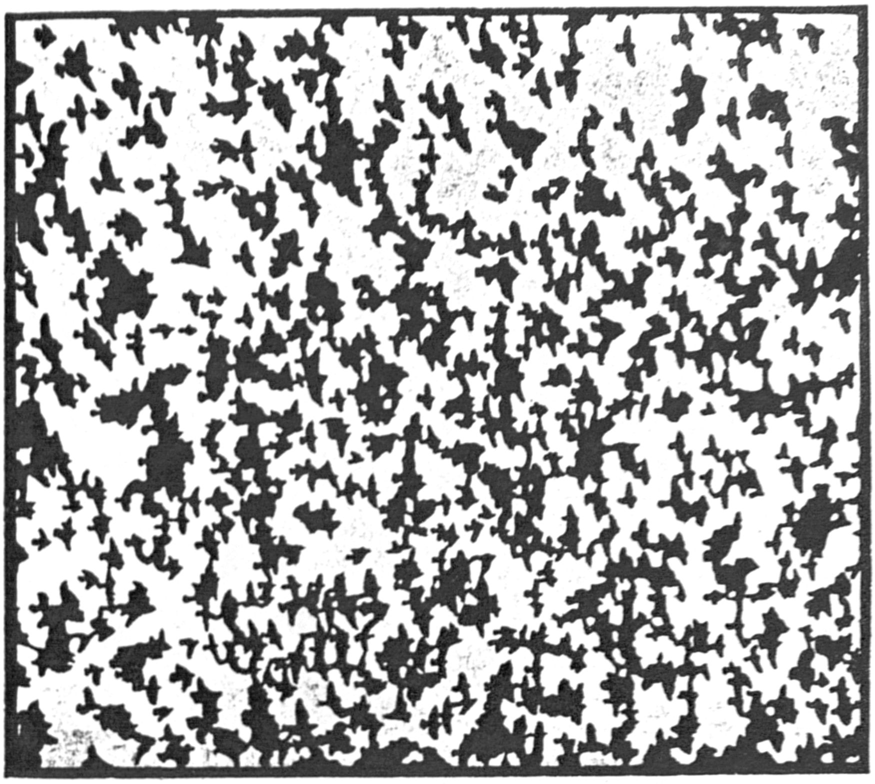


figure 81(b)

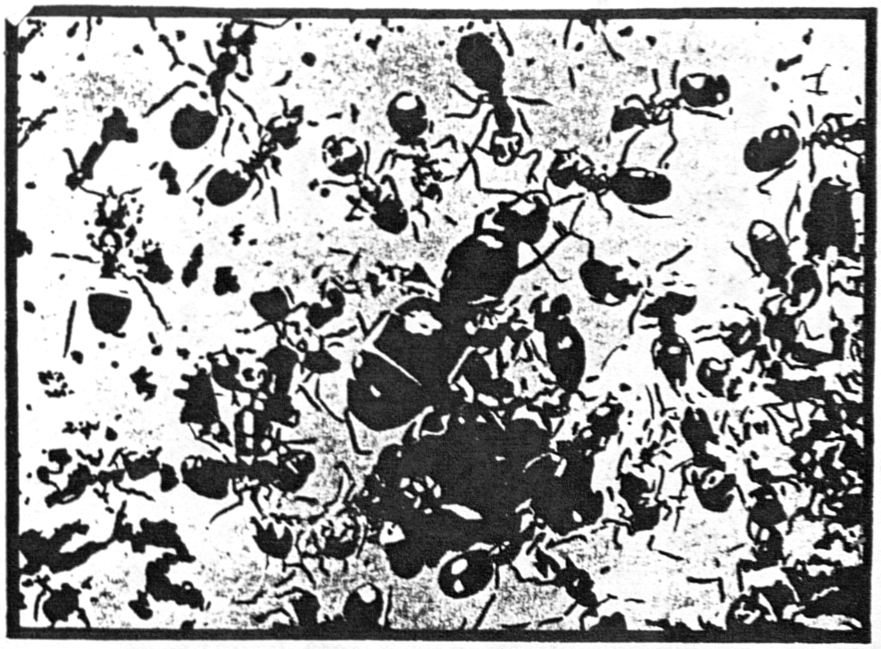
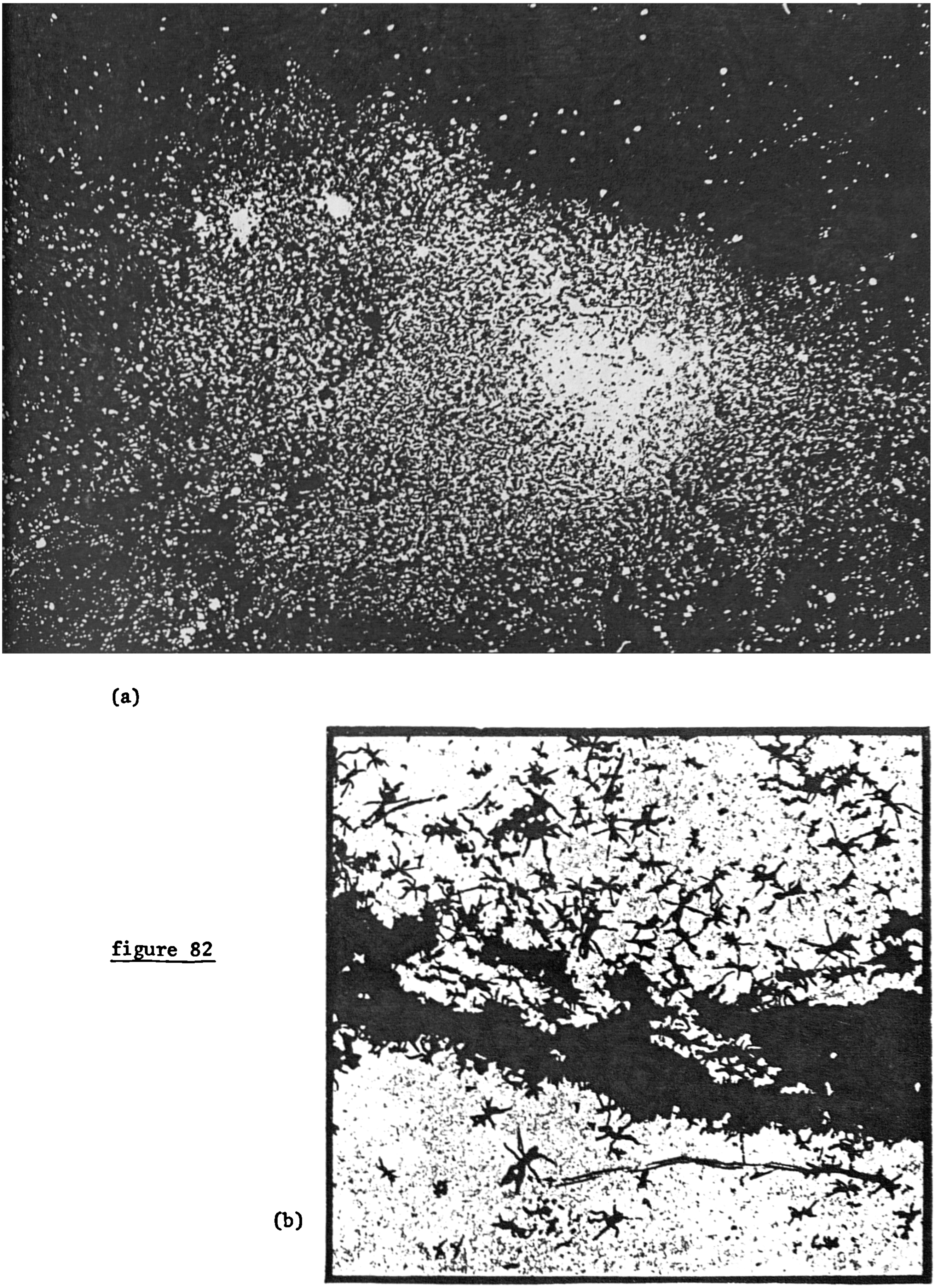


figure 81(c)

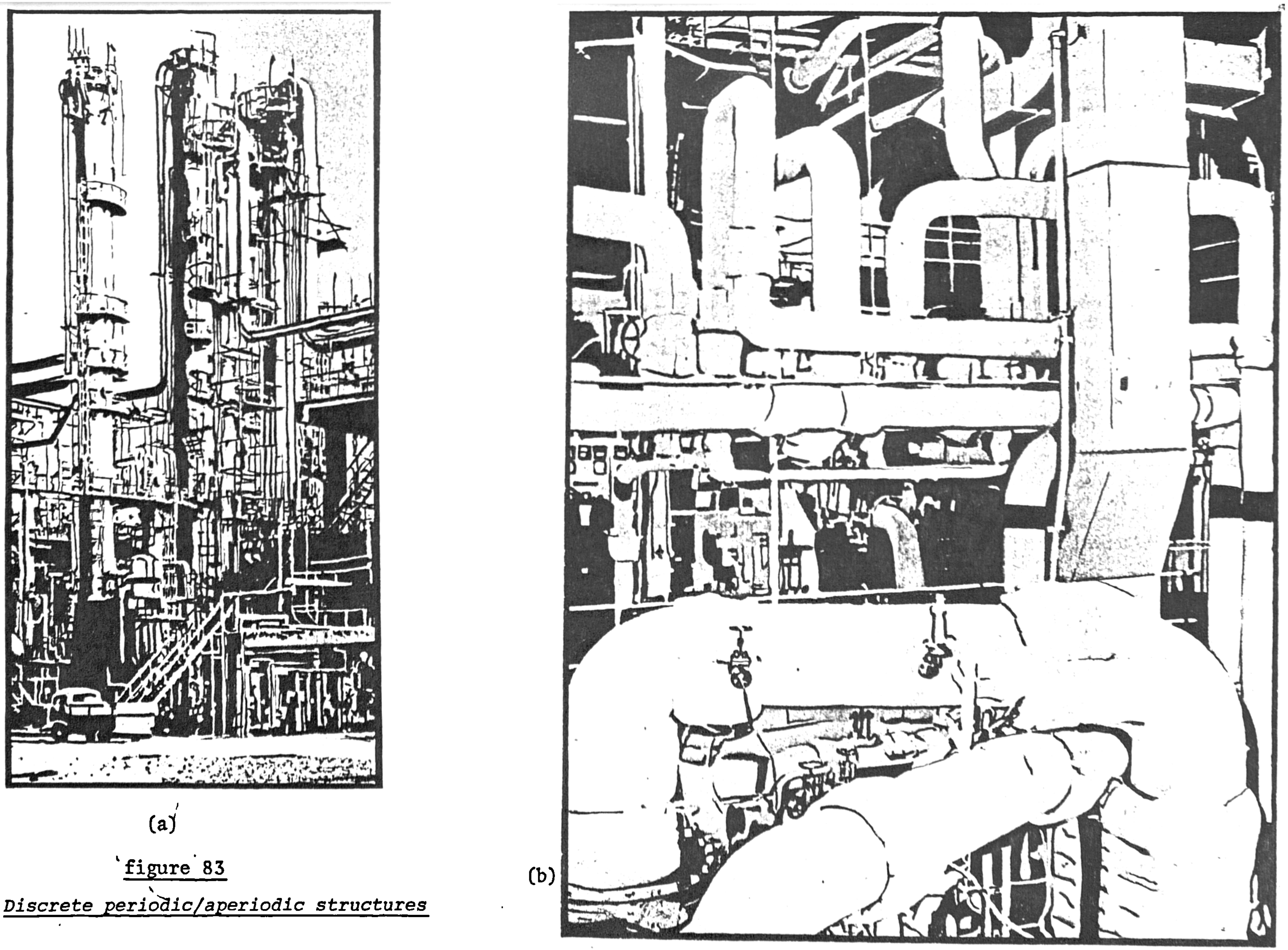


(a)

(b)

figure 82

Partially dispersed objects arranged within an overall directional constraint.



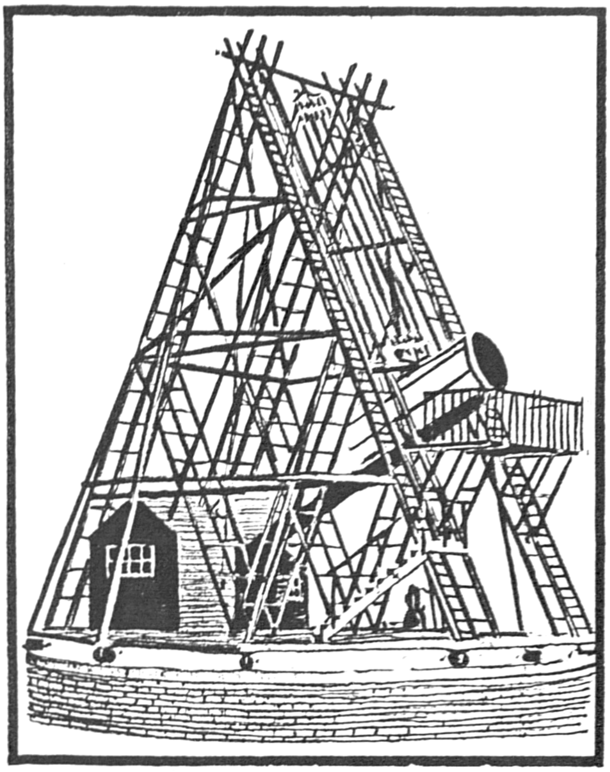
(a)

(b)

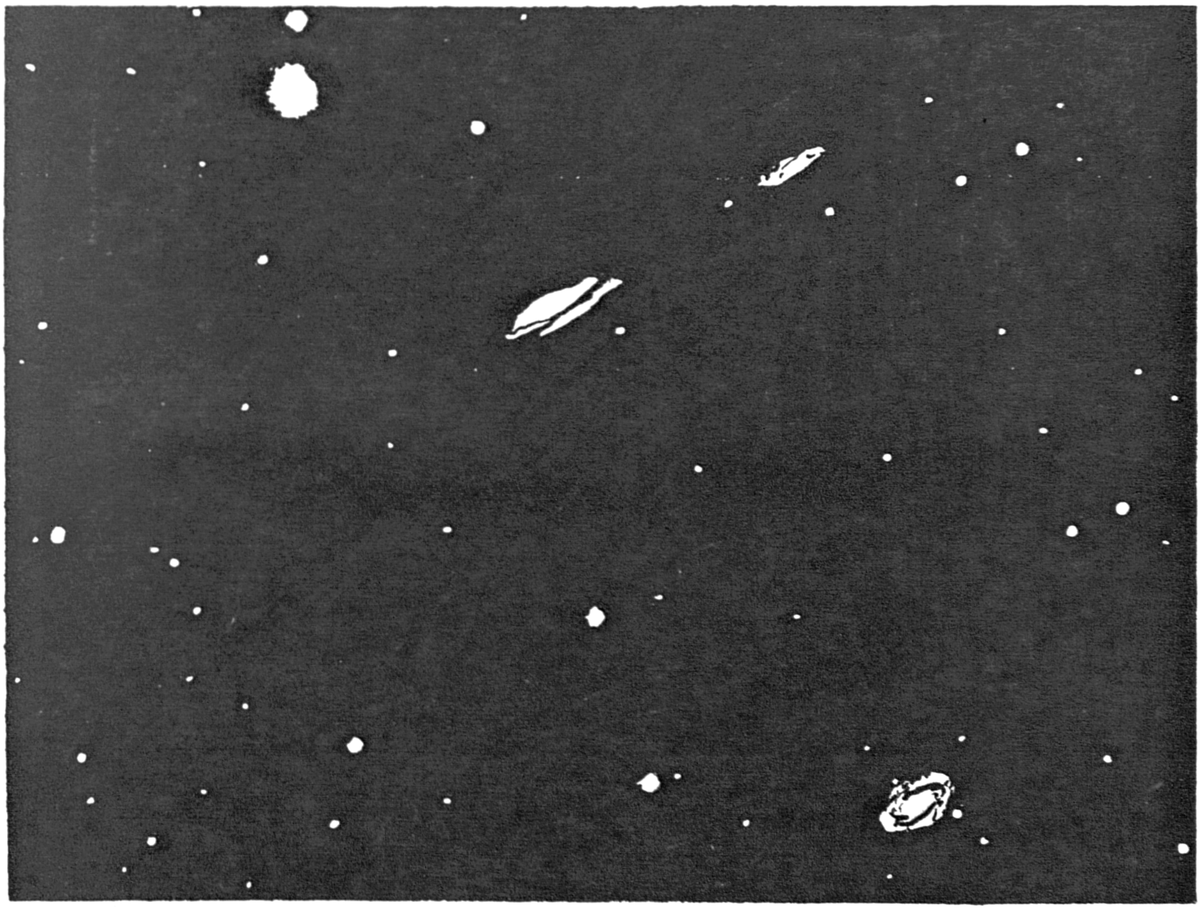
figure 83

Discrete periodic/aperiodic structures

figure 84



(a)



(b)

Criss-crossing web structures and isolated clusters.

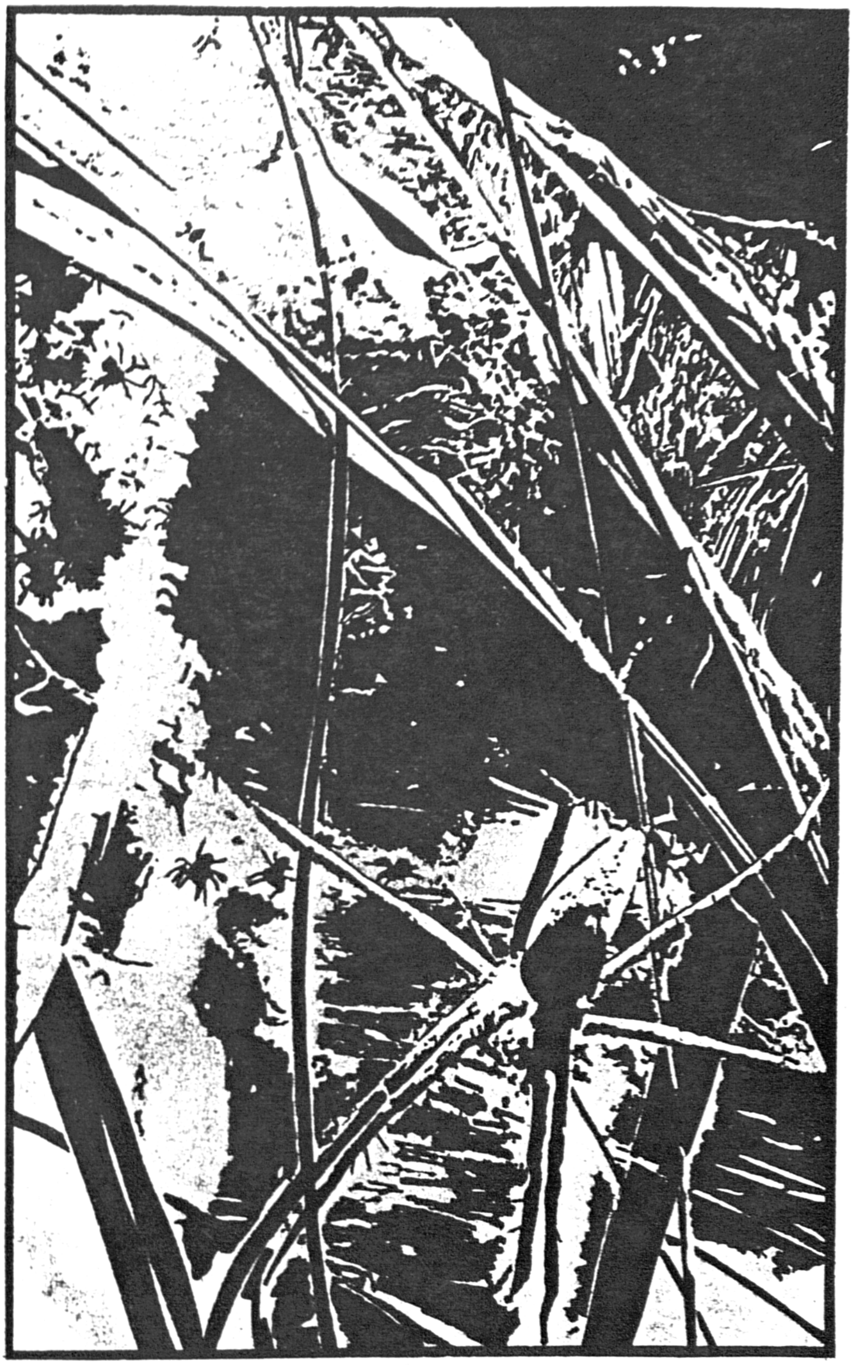


figure 84(c)

In performing *MACRICISUM* not only does the use of such images supply the required visual stimulation and thus help to maintain a high level of attention, but it also highlights aspects of the sound, giving a clearer perception of the structure and a better understanding of its relationship to natural patterns.

After listening to the piece at a public performance, the critic Peter Stadlen, writing in the *Daily Telegraph*,[1] considered it "wholly true to the medium" and recognised the distinguishing characteristics of each movement. He also found the visual aspect "particularly significant" and compared the process of selecting the images to "Beethoven going for a stroll after completing the Pastoral".

MACRICISUM was performed at the Skinners' Hall, London in July 1978, at the City University, London in March 1979 and at the Reid Concert Hall, Edinburgh in April 1980. The study *SKIRTRIKS* was broadcast on BBC Radio 3 in January 1980.

CHAPTER EIGHT

STOCHASTIC PATTERN GENERATION (III)

- STOCHASTIC GRAMMARS

The most potentially exciting and powerful extension of stochastic processes is in the area of formal grammars. Many musical structures have traditionally been represented in a form comparable to a simple grammar. For instance, a Minuet and Trio is described as having the form AABA, and a rondo as ABACADA. Recently, more formal and extended applications of grammars for defining musical structures have appeared.[1, 2] By extending stochastic systems to apply to formal grammars, a powerful generative tool is created.

The purpose of this chapter is not to present a complete theory of grammars for musical description, but to briefly explain the basic structure and power of formal grammars with simple illustrative examples, as a necessary prelude to the introduction of stochastic web grammars.

8.1 Definition of a string grammar

A grammar G is defined formally as a 4-tuple.

$$G = \langle V_N , V_T , P , S \rangle$$

where V_N is a set of *variables* or *non-terminals*. A variable is a structural marker which will be rewritten by something else,

V_T is a set of *terminals* which terminate any derivation and are not rewritten,

P is a set of *production rules* which specify how variables are to be rewritten by a combination of variables and terminals: a right-facing arrow, "→", is used in a production rule to specify that whatever is on the left of the arrow is to be replaced by whatever is on the right,

S is a *starting symbol* which begins the generative process. $S \in V_N$.

It follows that $V_N \cap V_T = \emptyset$, the null set. That is the set of variables and the set of terminals can have no members in common.

The set $A = V_N \cup V_T$ is known as the *alphabet* of the grammar.

A *string* is any sequence of symbols from the alphabet. A^* is the set of all possible strings.

A *terminal string* is a sequence of terminals generated by the grammar. The complete set of all possible terminal strings forms the *language* L generated by the grammar. $L \subseteq V_T^*$.

The number of symbols in a string is the *length* of the string.

Example 8.1.1

Consider the grammar G_1.

$$G_1 = \langle V_{N1} , V_{T1} , P_1 , A \rangle$$

where $V_{N1} = \{ A , B \}$

$V_{T1} = \{ a , b , c \}$

and P_1 consists of the following production rules:

$$A \rightarrow a B a$$

$$B \rightarrow b$$

$$B \rightarrow c$$

To generate a string, the starting symbol " A " is rewritten by the string " a B a ", then in this string, the non-terminal " B " is replaced by " b " or " c ". Thus the grammar can generate just two terminal strings, " aba " and " aca ", both of length three.

If A_1 is the alphabet of the grammar and L_1 is the language generated by the grammar then,

$$A_1 = \{ A , B , a , b , c \}$$

$$L_1 = \{ aba , aca \}$$

The powerful generative capacity of grammars becomes more evident when *recursive* production rules are used; that is, when the same variable occurs on both sides of a production rule.

Example 8.1.2

Consider the grammar G_2.

$$G_2 = \langle V_{N2}, V_{T2}, P_2, D \rangle$$

where $V_{N2} = \{A, B, C, D\}$

$V_{T2} = \{a, b, c\}$

and P_2 consists of the following production rules:

D	→	A C A	(1)
A	→	A B A	(2)
A	→	a	(3)
B	→	b	(4)
C	→	C C	(5)
C	→	c	(6)

A possible sequence of derivations is:

D →	ACA	(rule 1)
→	ABACCABA	(replacing both A's by ABA (rule 2) and C by CC (rule 5))
→	ABAbABACCCCaba	(replacing each of the first two A's by ABA (rule 2), each of the two C's by CC (rule 5), the remaining A's by a and B's by b (rules 3 and 4))
→	abababacccCCaba	(replacing all the A'a by a, B's by b, the first three C'c by c and the last C by CC)
→	abababaccccccaba	(replacing each of the remaining C's with c)

The alphabet A_2 of this grammar is $\{A, B, C, D, a, b, c\}$. Strings may be of any length, without any upper limit.

The language L_2 generated by G_2 is infinite, but it can be seen that this grammar will always generate an alternating sequence of a's and b's followed by a group of c's and followed again by a sequence of alternating a's and b's. The language may thus be represented in the following form.

$$L_2 = \{ (ab)^p\ a\ c^q\ (ab)^r\ a \mid p,r = 0,1,2,\ldots\ ;\ q = 1,2,\ldots \}$$

For many types of grammars it is possible to plot a tree diagram to show the derivation of a string through the sequence of production rules. Figure 85 shows the tree diagram for the string of example 8.1.2. The hierarchic derivation of the string is clearly evident in this figure.

The terminal string is equivalent to the event sequence generated by a Markov Chain and discussed in chapter six. The set of terminals can thus be mapped into an event space for the musical application of grammars.

Example 8.1.3

The elements a, b, and c in the terminal set of example 8.1.2 can be replaced by notes of the same name. The generated sequence will then transcribe into the form of figure 85 (page 227).

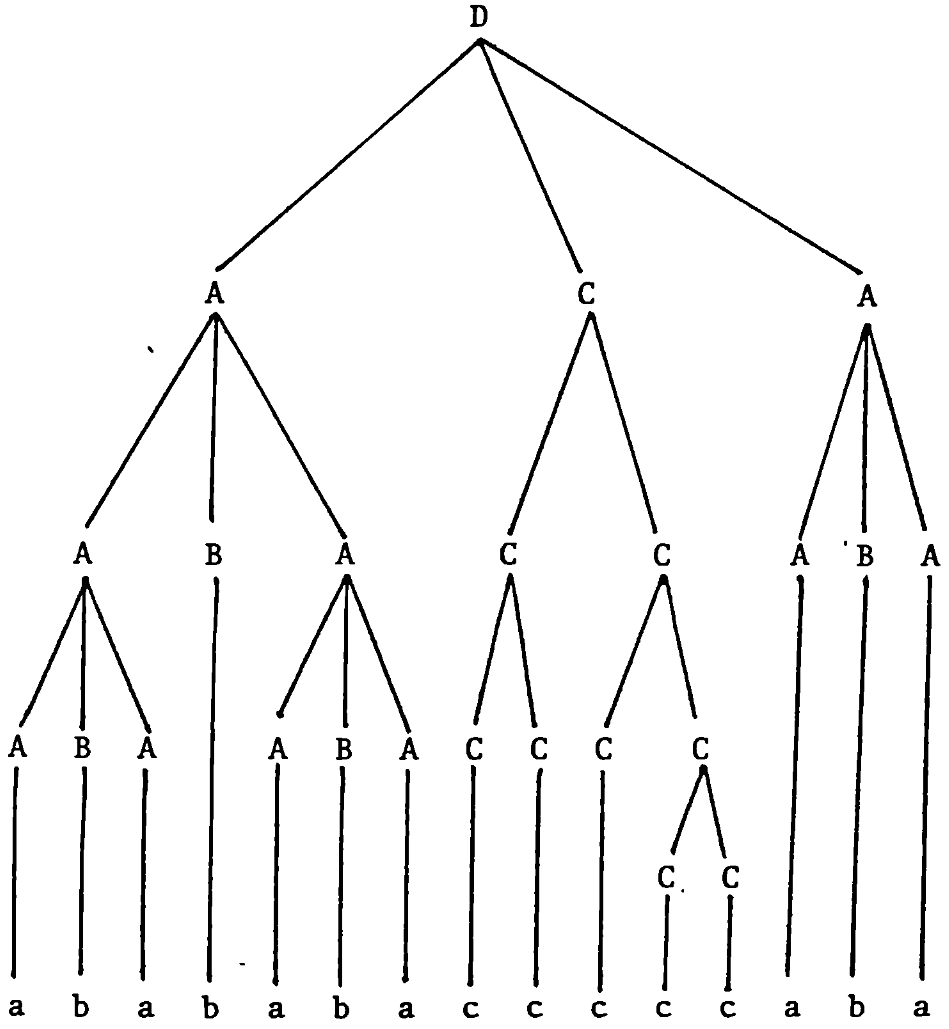


figure 85

Tree diagram of a grammar derivation

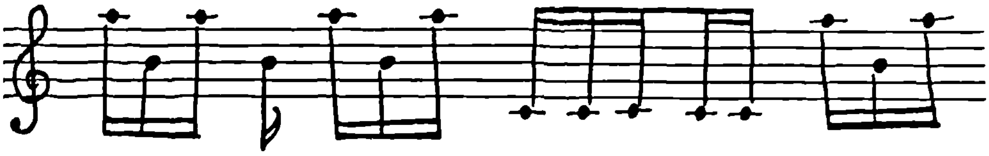


figure 86

Musical transcription of the terminal string of figure 85

It can be seen that the use of grammars to generate sequences can offer, in general, a much more powerful facility than the more restricted Markov Chain systems discussed earlier. The grammatical heirarchy preserved in the system of production rules means that each event in the generated sequence has a strong contextual function in the structure as a whole, and is not just related to its immediate predecessor. Left, right and embedded structures can be built up. Not only are such structures obviously paralleled in the system of natural languages, but similar patterns have also been identified by the 'structuralist school' in anthropological, social and literary studies.[3] This leads to the conclusion, vigorously championed by Chomsky,[4] that such structures closely emulate the natural thought processes of the human mind. Recent advances in formal linguistics applied to pattern recognition[5] and in the furtherance of artificial intelligence studies, have extended the techniques into almost every sphere of human endeavour including electronic, structural and chemical engineering, meteorology, strategic studies, medicine and economics.

The Markov Chain systems considered in chapter six are eminently suitable for certain musical uses as demonstrated in chapter seven. Successful practical applications have shown this to be the case. However, it will be demonstrated shortly the Markov Chains are in fact equivalent to a special type of grammar. Naturally a more general system offers greater scope for application, whereas specialist sub-systems may be suitable for specific tasks.

8.2 Types of string grammars

It is possible that a number of different grammars may all generate the same language, and any one grammar may be expressed in a number of ways with different variables and production rules. A basic classification devised by Chomsky separates grammars into four basic types depending upon the nature of the production rules.[6] The four types do not form independent classes of grammars, but each is a special case of the following type (in the order presented below). This classification has been universally adopted and is very useful.

8.2.1 Chomsky Type 3 - Finite State Grammar

A Type 3 grammar only allows productions of the form:

either $X \rightarrow y\,Z$ or $X \rightarrow Z\,y$

and $X \rightarrow y$

where $X, Z \in V_N$, $y \in V_T$.

Non-terminating productions must all be of the form $X \rightarrow y\,Z$, or all of the form $X \rightarrow Z\,y$. They may not be mixed in one grammar.

Such a grammar is known as a *finite-state*, or *regular grammar*. If productions are of the form $X \rightarrow y\,Z$ then it is also known as a *right-linear* grammar; and as a *left-linear* grammar if productions are of the form $X \rightarrow Z\,y$.

It can be seen that in a right-linear grammar any sequence of productions will build a string up linearly from left to right:

$$\begin{aligned} X &\rightarrow y_1\ Z_1 \\ &\rightarrow y_1\ y_2\ Z_2 \\ &\rightarrow y_1\ y_2\ y_3\ Z_3 \\ &\rightarrow y_1\ y_2\ y_3\ y_4\ Z_4 \\ &\rightarrow y_1\ y_2\ y_3\ y_4\ y_5\ Z_5 \\ &\quad \ldots\ldots\ldots\ \text{etc.} \end{aligned}$$

and so the tree diagram of a right-linear grammar will always have the form of figure 87.

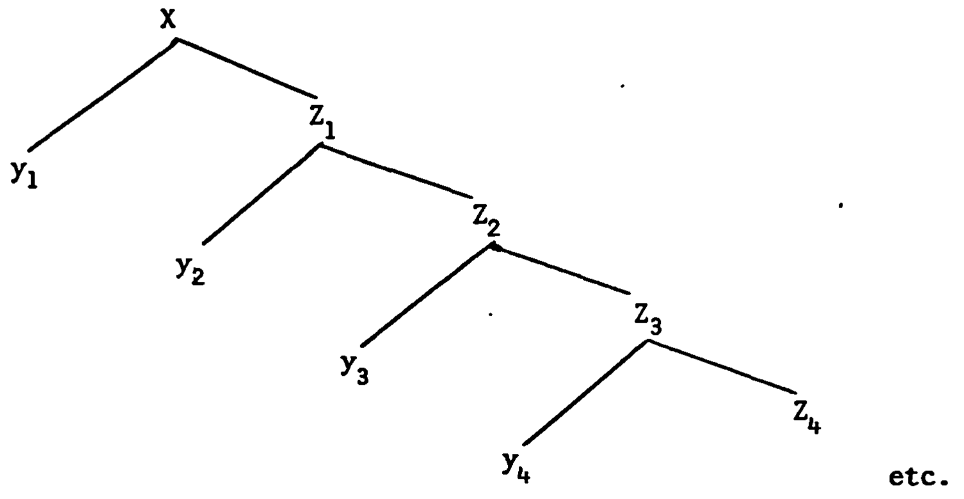


figure 87

The tree diagram of a set of finite-state grammar derivations

This generation process operates in precisely the same way as a Markov chain, with the identity of each element in the string being dependent only upon its immediate predecessor. It should therefore be evident that a Markov Chain is *structurally* equivalent to a Type 3 grammar. There are of course a few minor differences. A grammar allows for a specific start to a string and a specific end when a final production X → y is applied.

Example 8.2.1.1

Consider the Markov Chain of example 6.1.1 (page 96). The terminal set $\{ e_1, e_2, e_3 \}$ will be used to refer to the three events. The non-terminal E_i will be used when it is to be replaced by terminal e_i followed by another non-terminal.

The structure defined by the matrix:

$$\begin{bmatrix} \cdot 8 & 0 & \cdot 2 \\ 0 & 0 & 1 \\ \cdot 2 & \cdot 8 & 0 \end{bmatrix}$$

can be rewritten as the following set of productions:

$$\begin{array}{lcll} E_1 & \rightarrow & e_1 E_1 & (1) \\ E_1 & \rightarrow & e_1 E_3 & (2) \\ E_2 & \rightarrow & e_2 E_3 & (3) \\ E_3 & \rightarrow & e_3 E_1 & (4) \\ E_3 & \rightarrow & e_3 E_2 & (5) \end{array}$$

No starting symbol is specified by the Markov Chain. e_1 was used as the first event in the generated sequence of example 6.1.1, so E_1 may be arbitrarily adopted as the starting symbol in this case.

As they stand, the five production rules will never terminate a sequence of generations, but will continue to cycle indefinitely. To permit termination of a string, the following additional productions may be added.

$$\begin{array}{lcll} E_1 & \rightarrow & e_1 & (6) \\ E_2 & \rightarrow & e_2 & (7) \\ E_3 & \rightarrow & e_3 & (8) \end{array}$$

Only one of these productions need be added, if the sequence is required to end with a specific event.

An example of a possible sequence of generations is given.

$$\begin{array}{lll} E_1 & \rightarrow e_1\ E_1 & \text{(rule 1)} \\ & \rightarrow e_1\ e_1\ E_1 & \text{(rule 1)} \\ & \rightarrow e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ E_1 & \text{(rule 1 five times)} \\ & \rightarrow e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ E_3 & \text{(rule 2)} \\ & \rightarrow e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_3\ E_2 & \text{(rule 5)} \\ & \rightarrow e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_1\ e_3\ e_2\ E_3 & \text{(rule 3)} \\ & \ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{etc.} & \end{array}$$

It can be seen that precisely the same sequence is being generated as appears in example 6.1.1. This will eventually terminate with the application of rule 6.

The use of probabilities in production application is considered in the section on stochastic grammars below (8.4).

In such a simple case as the above example, the representation of a Markov Chain as a grammar seems cumbersome compared to the matrix representation. However, where a large terminal set or event space is involved, where a matrix with a large number of zero's may result, a grammar representation may be more efficient. this is the case with the actual representation in a computer program of a random walk. In the programs described earlier, statements related to the following production forms were used.

$$\begin{array}{lll} E_n & \rightarrow & e_n\ E_{n-1} \\ E_n & \rightarrow & e_n\ E_{n+1} \end{array}$$

8.2.2 Chomsky Type 2 - Context free grammar

A Type 2 grammar permits productions of the form:

$$X \rightarrow \alpha \qquad\qquad \text{where} \quad \alpha \in A^*.$$

Thus any variable, X, can in general be replaced by a string of terminals and variables.

It can be seen that a Type 3 grammar is a special case of a Type 2 grammar where

$$\alpha = x\,Y$$

$$\text{or} \quad \alpha = x \qquad\qquad (x \in V_T\,,\ Y \in V_N\,)$$

It is possible with a context free grammar to have greater control over the sequence generation, and to model structural patterns in production rule and non-terminal definitions.

Example 8.2.2.1

Consider the grammar G_3

$$G_3 = \langle V_{N3}\,,\ V_{T3}\,,\ P_3\,,\ B \rangle$$

where $V_{N3} = \{ B, C, M \}$

$V_{T3} = \{ a, b, c, d, g, d_1, d_2, g_1 \}$

and P_3 consists of the following productions:

$$B \rightarrow M\,M\,M\,g_1 \qquad (1)$$

$$M \rightarrow C\,C \qquad (2)$$

$$M \rightarrow g_1 \qquad (3)$$

$$C \rightarrow g\,a \qquad (4)$$

$$C \rightarrow b\,a \qquad (5)$$

$$C \rightarrow b\,c \qquad (6)$$

$$C \rightarrow d\,c \qquad (7)$$

$$C \rightarrow d_1 \qquad (8)$$

$$C \rightarrow d_2 \qquad (9)$$

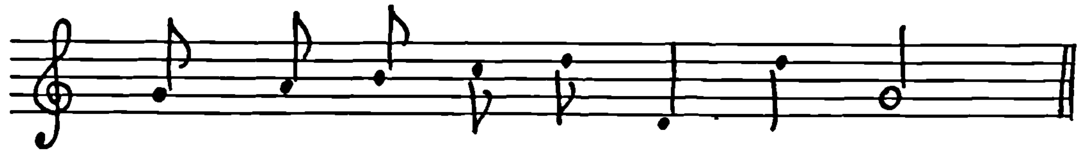
Of the non-terminals, M stands for notes adding up to a minim and C for notes adding up to a crotchet.

The musical equivalents of the terminal set V_{T3} are scored in figure 88 below.

The production rules guarantee the generation of two bars of four crotchet-beats, with a correct time structure, and which always finishes on a minim G.

A possible generation sequence may be as follows:

$$\begin{array}{lll} B & \rightarrow M\ M\ M\ g_1 & \text{(rule 1)} \\ & \rightarrow C\ C\ M\ M\ g_1 & \text{(rule 2)} \\ & \rightarrow C\ C\ g_1 C\ C\ g_1 & \text{(rules 3 and 4)} \\ & \rightarrow d\ c\ b\ a\ g_1 C\ C\ g_1 & \text{(rules 7 and 5)} \\ & \rightarrow d\ c\ b\ a\ g_1 d_1 b\ a\ g_1 & \text{(rules 8 and 5)} \end{array}$$



$V_{T3} = \langle g\ ,\ a\ ,\ b\ ,\ c\ ,\ d\ ,\ d_1\ ,\ d_2\ ,\ g_1 \rangle$

figure 88

A set of musical terminals

This generation may be represented by the tree diagram of figure 89 (page 235).

The terminal sequence will transcribe into the form of figure 90(a) (page 236).

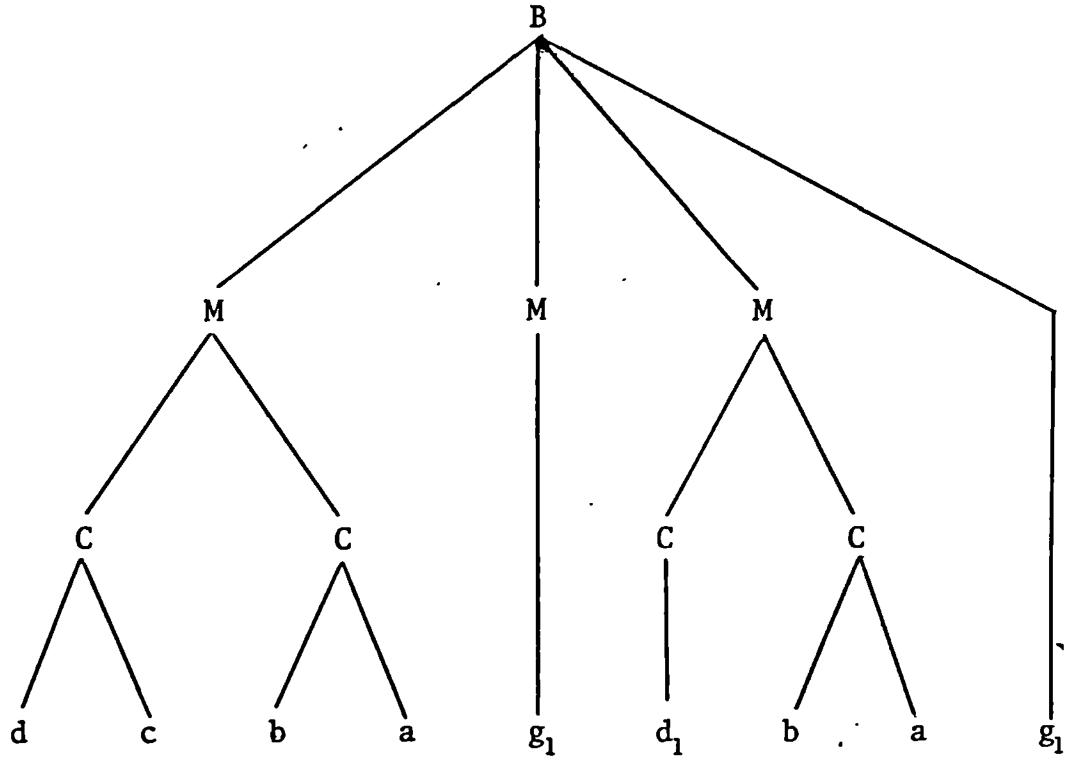


figure 89

The tree diagram of the derivation of a terminal string formed by applying a grammar over the terminal set of figure 88

Four other possible terminal strings are given:

g_1 b a d_2 d_1 d_2 g_1
b a b a d c d_2 b c g a g_1
d_2 d_2 g a b a g_1 g_1
d_1 d c d c b c d c b a g_1

These are transcribed in figure 90(b) - (e).

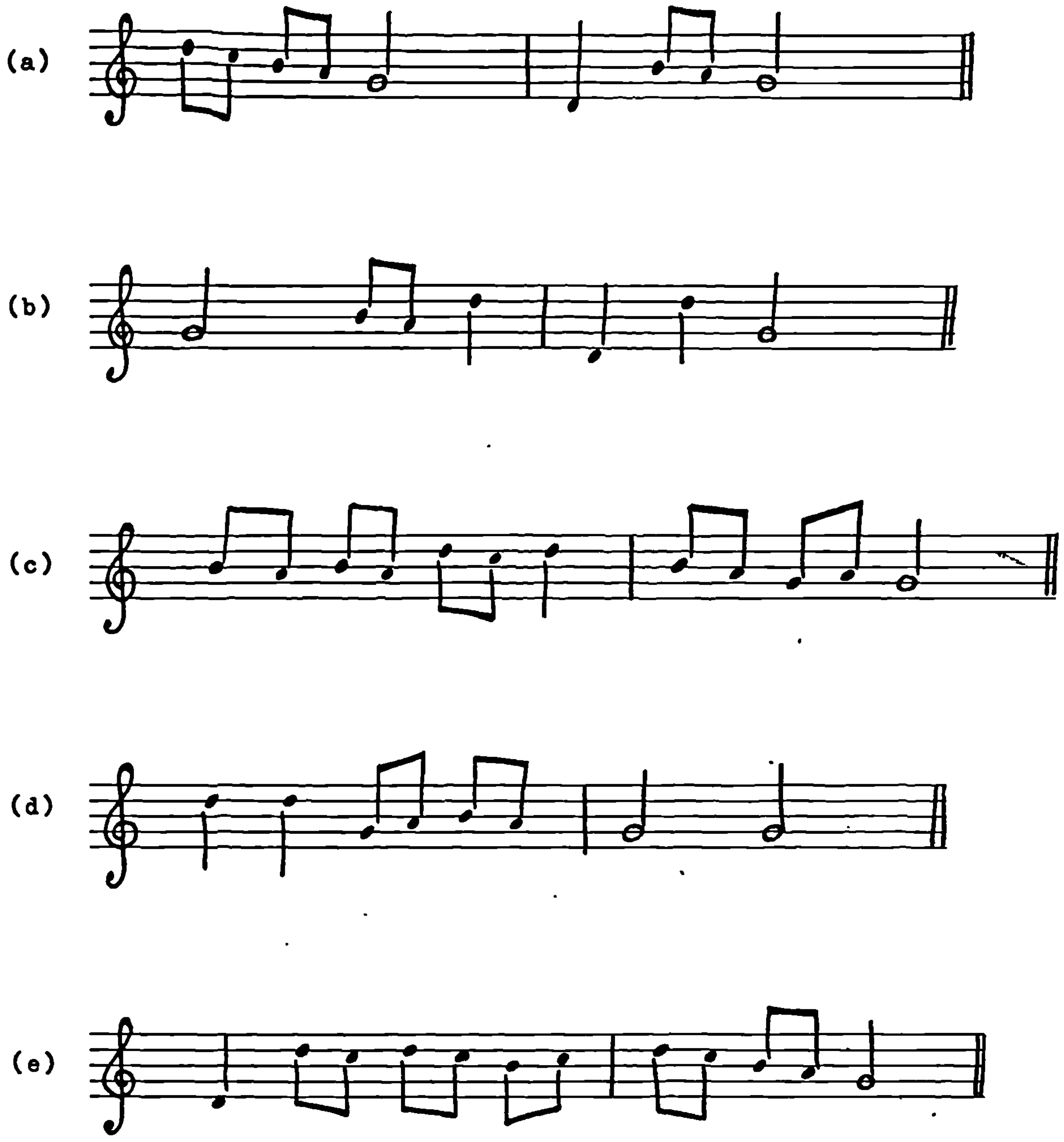


figure 90

Musical transcriptions of terminal event sequences generated by the same grammar as is illustrated in figure 89

This example will produce a *finite* set of terminal sequences. (There are nearly 50 000 possibilities). Any finite language can be generated by a finite state, Type 3 grammar, since it is always possible to set up a sequence of production rules to generate each possible terminal string individually. However, the hundreds of thousands of non-terminals and production rules that may be required render such an operation quite useless whereas the context free representation preserves the essential structural relationships. This is shown particularly clearly in the above example (figure 89) where the non-terminals express the time values and groupings.

A powerful, generative property of a context free grammar is its ability to handle *self-embedded* structures using recursive production rules where:

$$X \rightarrow \alpha X \beta \qquad (\alpha , \beta \in A^{*} , \ X \in V_N)$$

Grammars of this form will always produce an *infinite* set of terminal sequences.

Example 8.2.2.2

Consider the grammar G_4 .

$$G_4 = \langle V_{N4} , V_{T4} , P_4 , A \rangle$$

where $V_{N4} = \{A , B\}$

$V_{T4} = \{a , b\}$

and P_4 consists of the three productions:

$$A \rightarrow B\,A\,B$$
$$A \rightarrow a$$
$$B \rightarrow b$$

This grammar will produce a language of the form:

$$L_4 = \{ b^n a b^n \mid n = 0, 1, ... \}$$

where all strings consist of a single 'a' sandwiched between strings of 'b's *always of equal length.* A Type 3 grammar cannot cope with this sort of symmetry.

This feature of context free grammars makes them an ideal general tool for modelling the nested-motivic formations of musical structures at all levels. As will be demonstrated, this applies not only to traditional aspects of form, but also to the structure of actual sound textures and timbres.

8.2.3 Chomsky Type 1 - Context-sensitive Grammars

A Type 1, context sensitive grammar permits productions of the form:

$$\theta X \delta \rightarrow \theta \alpha \delta$$

where $\alpha, \delta, \theta \in A^*$ and $X \in V_N$.

This means that X is rewritten as α in the context of $\theta...\delta$ and accounts for the name *context sensitive* grammar. This also explains the use of the term *context free* for Type 2 grammars. A Type 2 context free grammar is a special case of a Type 1 context sensitive grammar.

The extra generality of context sensitive grammars does not offer any particular advantages in the type of application under consideration. In order to apply a context sensitive production, it is necessary to search through a string of terminals and non-terminals at each stage of generation, to find appropriate contexts for further generations. Although useful in some types of natural language processing, such productions can blur the structural elegance which is characteristic of Type 2 grammars.

8.2.4 Chomsky Type 0 - Unrestricted Grammar

In this, the most general of string grammars, any type of generation is permitted:

$$\alpha \rightarrow \beta \qquad (\alpha , \beta \in A^{*})$$

Such a generality means that it is possible for a production to shorten a string, re-arrange the order of elements in a string, or even cause a string to disappear completely! In fact, it would be possible to have a complex system of production rules defining an elaborate formal structure which led to the generation of a null terminal sequence; a *phantom macro-structure*. In musical terms, this would be an elaborately structed silence lasting for zero seconds! Although a certain brand of composer may be intrigued by such a possibility, its practical use is obviously nil.

The four grammatical types which have been identified form a structural heirarchy where each class of grammars of Type n includes Types greater than n. At the top of the hierarchy is Type 0 with no restrictions. At the bottom is Type 3 with most restrictions. A summary of the grammar Types and their relationships is given in figure 91.

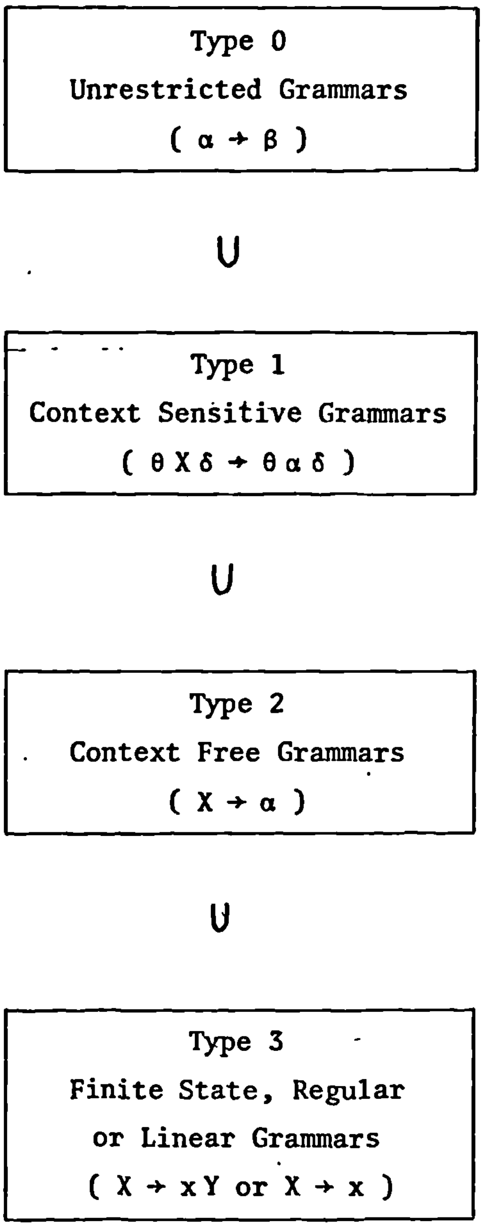

Type 0
Unrestricted Grammars
$(\alpha \rightarrow \beta)$
$\cup$
Type 1
Context Sensitive Grammars
$(\theta X \delta \rightarrow \theta \alpha \delta)$
$\cup$
Type 2
Context Free Grammars
$(X \rightarrow \alpha)$
$\cup$
Type 3
Finite State, Regular
or Linear Grammars
$(X \rightarrow xY$ or $X \rightarrow x)$

figure 91

The hierarchical relationships between the four 'Type' grammars

8.3 Web Grammars

The grammars described above are all *string* grammars, which generate a single linear sequence of elements. In musical structures, however, as well as the horizontal relationships occuring between sound in a time series, there are also vertical relationships which govern their simultaneous arrangement. It is for such situations that the use of web grammars is suggested.

In a string grammar, productions are always of the form:

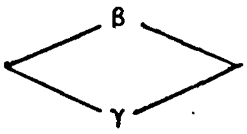
$$\alpha \rightarrow \beta\gamma$$

This specifies that α will be rewritten as β followed by γ. This assumes the existance of a redundant operator in between β and γ to specify concatenation of the two strings. The symbol "." could have been used, but it is not necessary and the operation is clearly understood. In a web grammar, an additional operator "/" is introduced so that productions may also be of the form:

$$\alpha \rightarrow \beta / \gamma$$

This means that α is replaced by string β *at the same time as* γ. In order to avoid ambiguity, the new string must be enclosed in brackets when written as part of a string sequence on one line.

Alternatively, the relationship may be represented in the two-dimensional plane by using a *web diagram*, where

$\alpha \rightarrow$ 

Example 8.3.1

Consider the grammar G_{W1} .

$$G_{W1} = \langle V_{N5} , V_{T5} , P_5 , A \rangle$$

where $V_{N5} = \{ A , B , C \}$

$V_{T5} = \{ a , b , c , d , e \}$

and P_5 consists of the following productions:

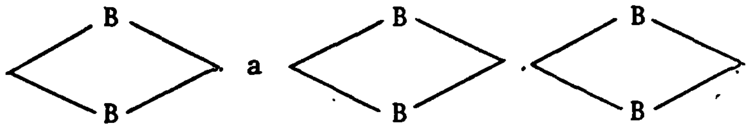
$$\begin{array}{lllr} A & \rightarrow & A\,A & (1) \\ A & \rightarrow & B/B & (2) \\ A & \rightarrow & a & (3) \\ B & \rightarrow & C\,C & (4) \\ B & \rightarrow & b & (5) \\ C & \rightarrow & c & (6) \\ C & \rightarrow & d/e & (7) \end{array}$$

The following is a possible generation sequence.

$A \rightarrow A\,A\,A\,A$ (rule 1 three times)

$\rightarrow (B/B)\ a\ (B/B)\ (B/B)$ (rules 2, 3, 2 and 2)

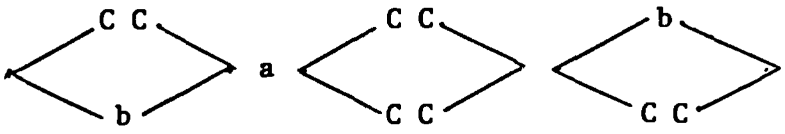
This may be represented:



Continuing the generation gives:

$\rightarrow (C\,C/b)\ a\ (C\,C\,/\,C\,C)\ (b/C\,C)$ (rules 4 and 5)

which may be represented:

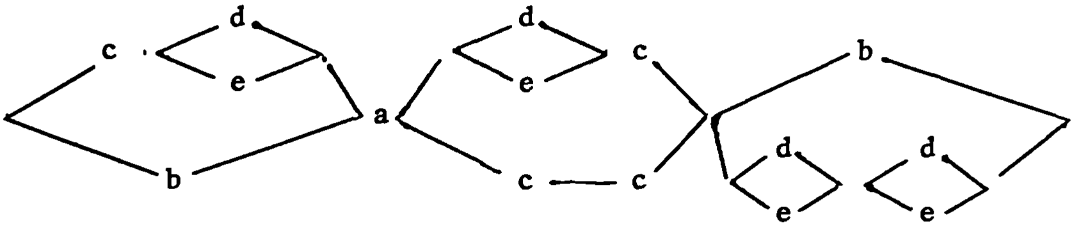


Terminating the generation gives:

$$\rightarrow \quad ((c\ (d/e))\ /\ b)\ a\ (((d/e)\ c)\ /\ c\ c)\ (b\ /((d/e)\ (d/e)))$$

(rules 6 and 7)

which may be represented:



The use of such grammars has been found particularly useful in pattern recognition applications, for describing and analysing images in the two-dimensional plane.[7,8] Similarly, as pointed out above, they are eminently suitable for representing musical structures. The following grammar will generate simple passages in two parts.

Example 8.3.2

Consider the web grammar G_{W2}.

$$G_{W2} = \langle V_{N6}, V_{T6}, P_6, B \rangle$$

where $V_{N6} = \{B, C, L_1, M_1, L_2, M_2\}$

$V_{T6} = \{b, c, d, s, g_1, c_1, d_1, s_1, c_2, d_2, s_2, g_2, g_3\}$

and P_6 consists of the following production rules:

$B \rightarrow B\,B$

$B \rightarrow M_2/L_2$

$B \rightarrow M_1/L_1$

$M_1 \rightarrow C_1\,C_1$

$M_1 \rightarrow g_2$

$C_1 \rightarrow d\,c$

$C_1 \rightarrow b\,c$

$C_1 \rightarrow d_1$

$C_1 \rightarrow g_1$

$C_1 \rightarrow b_1$

$L_1 \rightarrow g_3$

$L_1 \rightarrow g_2$

$L_1 \rightarrow g_1\,d_2$

$L_1 \rightarrow g_1\,g_1$

$M_2 \rightarrow s_1\,c_1$

$M_2 \rightarrow s\,d\,c_1$

$M_2 \rightarrow c\,d\,s$

$M_2 \rightarrow c_1\,g_1$

$L_2 \rightarrow c_2\,s_1$

$L \rightarrow g_2$

An abbreviated possible generation sequence is given:

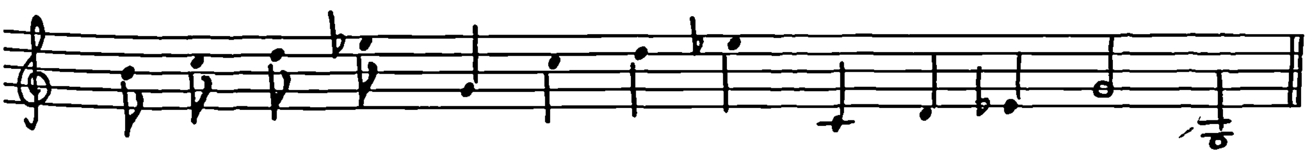
$$\begin{aligned} B &\rightarrow B\,B\,B\,B \\ &\rightarrow (M_2/L_2)\ (M_2/L_2)\ (M_1/L_1)\ (M_1/L_1) \\ &\rightarrow ((s_1\,c_1)/g_2)\ ((s\,d\,c_1)/(c_2\,s_1))\ (g_2/(g_1\,d_2))\ ((b\,c\,b\,c)/(g_1\,g_1)) \end{aligned}$$

With the terminal event space defined as in figure 92, the terminal sequence will transcribe into the form of figure 93(a).

Another possible terminal sequence is:

$$((d\,c\,b\,c)/(g_1 d_2))\ ((d_1 g_1)/g_2)\ ((s\,d\,c_1)/(c_2 s_2))\ ((b\,c\,d_1)/(g_1 d_2))$$
$$((c\,d\,s_1)/g_2)\ ((d_1 b_1)/g_3)$$

which transcribes into the form of figure 93(b).



$V_{T6} = b, c, d, s, g_1, c_1, d_1, s_1, c_2, d_2, s_2, g_2, g_3$

figure 92

A set of terminal events

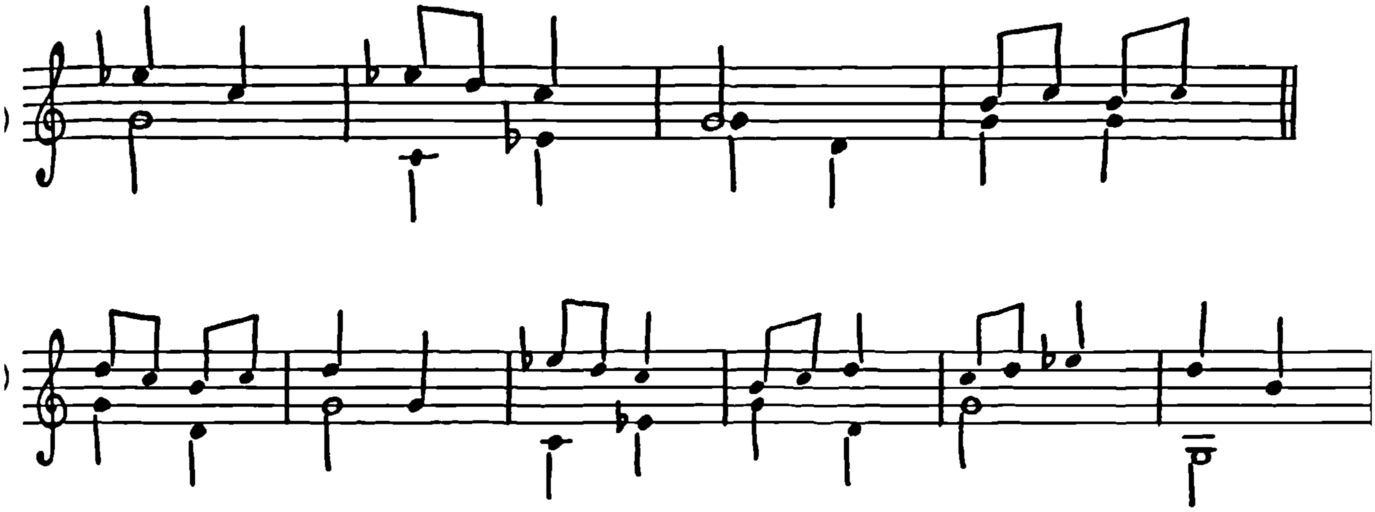


figure 93

Two possible sequences formed by applying a web grammar over the event space of figure 92

As has been emphasized many times before, there is no reason why the terminal event space need be limited to traditional 'notes'. Such an interpretation is convenient for presenting explanatory examples on paper to clarify structural issues, but the powerful implications of the wider scope of application, in defining the form of electronic music scores, for example, should be very evident. Further examples are given when stochastic grammars are discussed.

A string grammar may be considered to be a special case of a web grammar. It is possible to have web grammars of each of Chomsky's Types identified in the previous section. Their relationship may be represented by figure 94

The two-dimensional web grammar concept may be extended into n dimensions. A possible way of notating this is to use indexed operators, $/_1$, $/_2$, ... etc.. However, their usefulness is limited by the simple difficulty in conceptualizing structures built up in this way. Figure 95 (page 248) suggests a possible representation for a hypothetical terminal structure defined in three dimensions. More examples concerning the application and representation of web grammars appear in the following section.

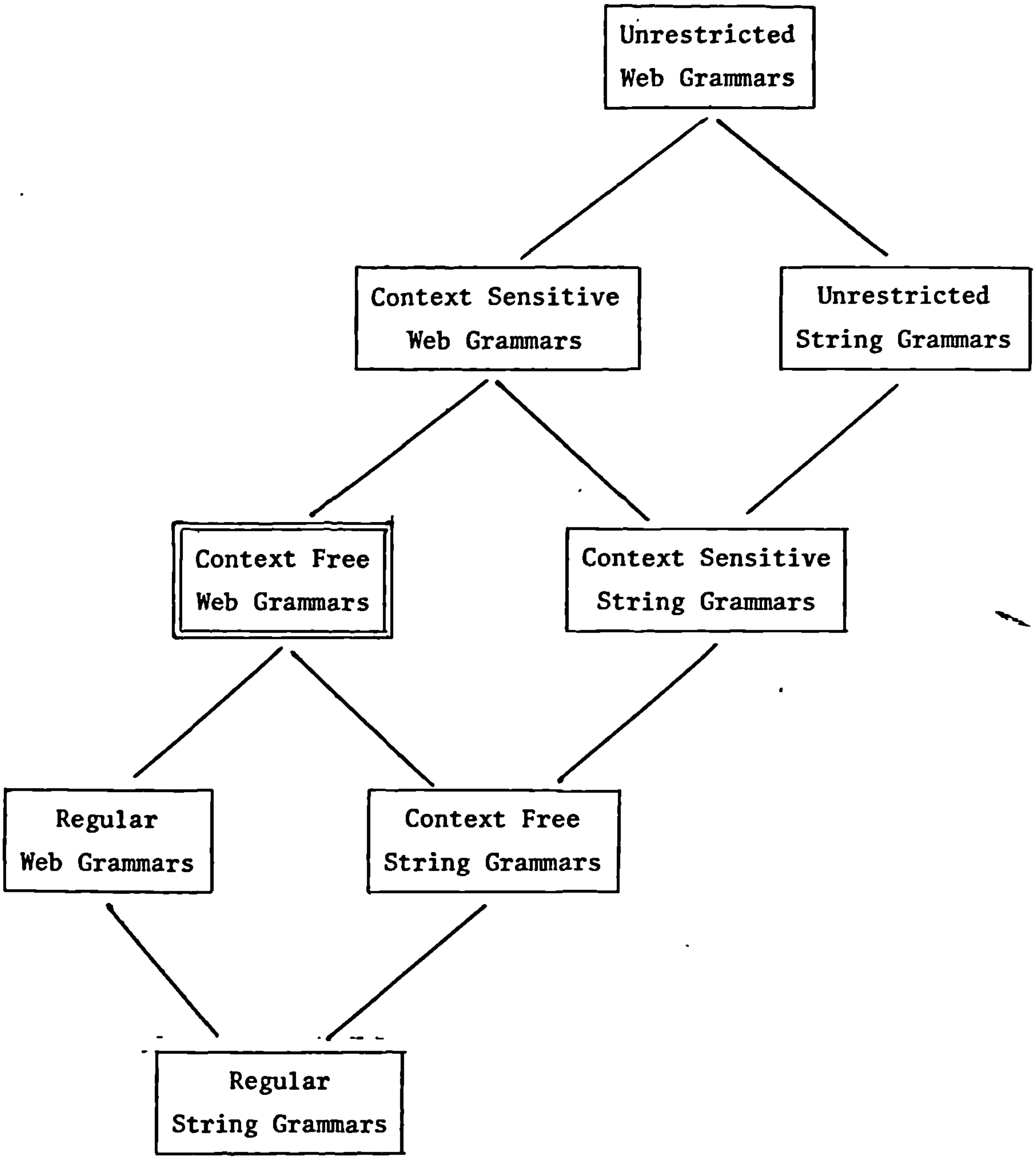

Unrestricted Web Grammars
Context Sensitive Web Grammars
Unrestricted String Grammars
Context Free Web Grammars
Context Sensitive String Grammars
Regular Web Grammars
Context Free String Grammars
Regular String Grammars

figure 94

The hierarchical relationships between types of string and web grammars

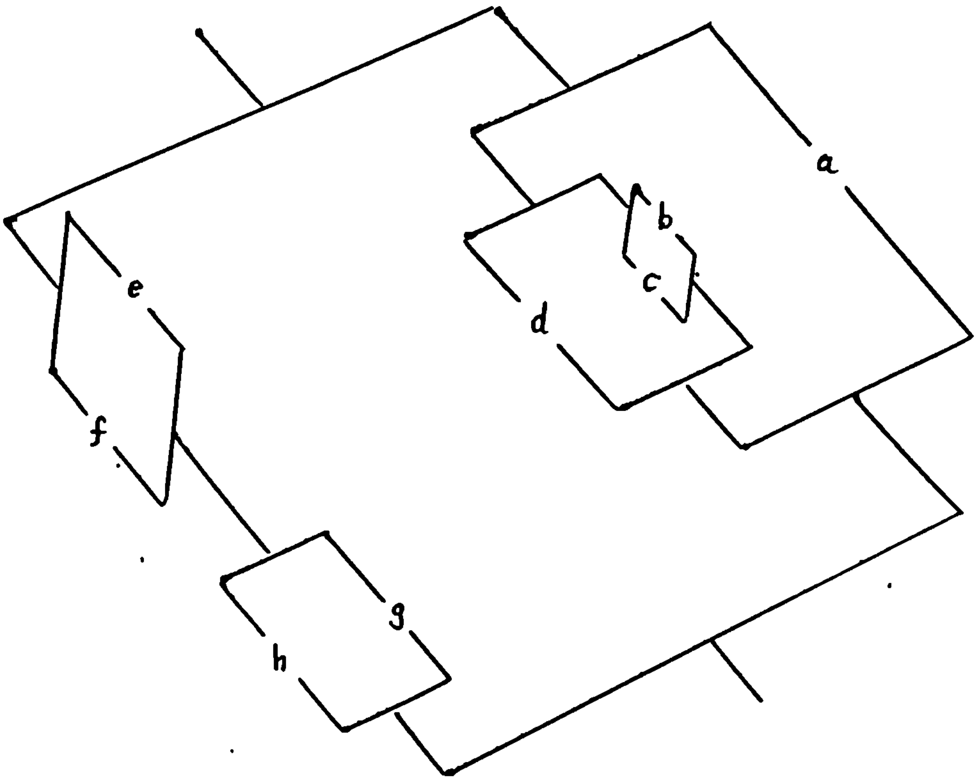


Figure 95

A representation of a terminal structure defined in three dimensions

8.4 Stochastic Grammars

The grammars described above are systems for description of generative structures. However, when used to derive actual sequences, no provision has been made for specifying which production rules whould be used when there is a choice of alternative strings for rewriting any non-terminal. To supply a means to control the choice of generations a *stochastic grammar* may be used.

8.4.1 Definition of a stochastic grammar

A stochastic grammar G_S is defined formally as a 5-tuple

$$G_S = \langle V_N , V_T , P , S , D \rangle$$

where V_N, V_T and S are defined as before (see 8.1).
P now becomes an *ordered* set of production rules and
D is a *probability assignment* over P.

$$|D| \text{ must equal } |P| = n$$

and the sum of the probabilities p_i corresponding to the set of production rules for each variable must equal unity, hence:

$$\sum_{i=1}^{n} p_i \in D = |V_N|$$

Example 8.4.1.1

Consider the grammar given in example 8.2.1.1 (page 231) to correspond to the Markov Chain of example 6.1.1 (page 96). The original matrix had the following form:

$$\begin{bmatrix} \cdot 8 & 0 & \cdot 2 \\ 0 & 0 & 1 \\ \cdot 2 & \cdot 8 & 0 \end{bmatrix}$$

and the derived set of production rules, $\mathcal{P}$, consisted of the following:

$$\begin{array}{lcll} E_1 & \rightarrow & e_1 E_1 & (1) \\ E_1 & \rightarrow & e_1 E_3 & (2) \\ E_2 & \rightarrow & e_2 E_3 & (3) \\ E_3 & \rightarrow & e_3 E_1 & (4) \\ E_3 & \rightarrow & e_3 E_2 & (5) \end{array}$$

It is now possible to form the probability assignment $\mathcal{D}$ over $\mathcal{P}$.

$$\mathcal{D} \quad = \quad (\cdot 8 \, , \cdot 2 \, , 1 \, , \cdot 2 \, , \cdot 8 \,)$$

In order to fit in a rule to terminate a generation, a little fiddling is necessary. Only rule 6, from the additional list in example 8.2.1.1, will be added to permit termination on event e_1 . To keep all the production rules with the same left hand side together, they will be renumbered as below.

$$\begin{array}{lcll} E_1 & \rightarrow & e_1 E_1 & (1) \\ E_1 & \rightarrow & e_1 E_3 & (2) \\ E_1 & \rightarrow & e_1 & (3) \\ E_2 & \rightarrow & e_2 E_3 & (4) \\ E_3 & \rightarrow & e_3 E_1 & (5) \\ E_3 & \rightarrow & e_3 E_2 & (6) \end{array}$$

The probability assignment $\mathcal{D}$ must also be reconstructed. Rule 3 will be given a fairly small probability to permit a reasonable length of sequence to be generated. The probabilities associated with rules 1 and 2, having the same non-terminal E_1 on the left hand side, then have to be adjusted so that the total sum of p_1, p_2 and p_3 equals one. Thus the assignment $\mathcal{D}$ may become:

$$\mathcal{D}' \quad = \quad (\cdot 78 , \cdot 18 , \cdot 04 , 1 , \cdot 2 , \cdot 8)$$

Notice that still,

$$\sum_{i=1}^{6} p_i \in \mathcal{D}' \quad = \quad 3 \quad = \quad | V_N |$$

The idea of a stochastic grammar and a web grammar can be combined to form a stochastic web grammar. Figure 96 represents the relationships between these different types of grammars. As with ordinary web and string grammars, it is possible to have stochastic grammars and stochastic web grammars of each of Chomsky's four Types, though it is not possible to represent their structural relationships in a diagram on the two-dimensional plane. In general, any subsequent reference to a stochastic grammar can be taken to mean a context-free stochastic web grammar.

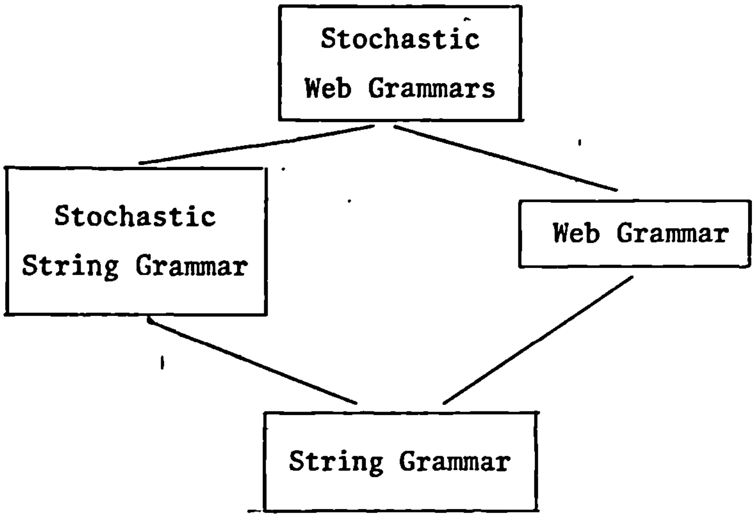


figure 96

The hierarchical relationships between stochastic, web and string grammars

8.4.2 Problem of Consistency

It is possible to interpret Regular Stochastic Grammars, such as the one in example 8.4.1.1 above, using the Markov Chain theory developed in chapter six. With other types of stochastic grammar, however, attempts to analyse the structure can become quite complex. One particular problem, of great practical significance, is whether any stochastic grammar will be at all likely to result in the termination of a sequence of generations. If the probabilities are badly defined, it is possible that a sequence of productions may never terminate and will continue to generate *ad infinitum*.

Example 8.4.2.1

Consider the stochastic grammar G_{S1} .

$$G_{S1} = \left\langle V_{N7} , V_{T7} , P_7 , A , D_7 \right\rangle$$

where $V_{N7} = \{A , B\}$

$V_{T7} = \{a , b\}$

P_7 consists of the following productions:

$$A \rightarrow A\ B\ A \qquad (1)$$
$$A \rightarrow a \qquad (2)$$
$$B \rightarrow b \qquad (3)$$

and $D = (\cdot 6 , \cdot 4 , 1)$

A possible generation sequence will be attempted:

Generation 1:	A → A B A	(rule 1)
Generation 2:	→ A B A b a	(rules 1, 3 and 2)
Generation 3:	→ A B A b A B A b a	(rules 1, 3 and 1)
Generation 4:	→ A B A b a b A B A b a b a	(rules 1, 3, 2, 1, 3, 2)

It can be seen from inspecting D that rule 1 can be expected to be applied three times for every two applications of rule 2. It should be evident from following the generation sequence begun above, that unless termination occurs in the first or second generation, the sequence will almost certainly continue indefinitely, as groups of non-terminals will appear in each generated sequence more frequently than terminals.

The probability of terminating in the first generation is simply the probability of applying rule 2, that is ·4 . The probability of terminating after two generations by applying rules 1, 2 and 2 is $(\cdot 6 \times \cdot 4 \times \cdot 4)$ which equals $\cdot 096$. After that the probability of termination plunges down deeply to rapidly approach zero.

In the above example, it is fairly easy to see how the generations will proceed and to spot that termination may be highly unlikely. But even with grammars that are slightly more sophisticated, this may be rather more difficult. The author has found in some experiments that a grammar became stuck in a seemingly endless series of generations. On running a computer program modelling such a grammar, computer time ran out before any results were produced. Subsequent inspection revealed that the probabilities were unrealistic or *inconsistent*. (See example 8.4.3.1 in the following section).

A general test for *consistency* of stochastic grammars has been developed.[9] To begin, a *first-moment matrix* M, similar to a stochastic matrix, is set up. Its rows and columns correspond to the non-terminals in V_N. Each entry m_{ij} in the matrix is calculated by adding together the probabilities associated with each production rule in which variable V_i is replaced by a sequence including V_j. If V_j occurs more than once in one production, then the probability associated with that production is multiplied by the number of times V_j appears in it.

Having obtained the matrix M, its eigenvalues are calculated. To find the eigenvalues, the matrix $(M - \lambda I)$ is formed, where I is the identity matrix and λ is a scalar quantity.

Thus if M has entries m_{ij}, then $M - \lambda I$ has the form:

$$\begin{bmatrix} m_{11}-\lambda & m_{12} & \cdot & \cdot & \cdot & m_{1n} \\ m_{21} & m_{22}-\lambda & \cdot & \cdot & \cdot & m_{2n} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ m_{n1} & m_{n2} & \cdot & \cdot & \cdot & m_{nn}-\lambda \end{bmatrix}$$

The eigenvalues (also called characteristic roots) are the solutions $\lambda_1, \ldots, \lambda_n$ to the equation formed by setting the determinant of this matrix to zero. If λ is the *principal eigenvalue*, that is the eigenvalue with the largest magnitude, then the grammar G_S is *consistent* if

$$|\lambda| < 1$$

and *inconsistent* if

$$|\lambda| > 1$$

where $|\lambda|$ is the absolute value of λ.

Example 8.4.2.2

Consider the grammar G_{S1} of example 8.4.2.1 above. "A" can be replaced by itself twice in a production with an overall probability of $0{\cdot}6$. Thus the entry m_{11} in the first moment matrix M_1 of the grammar will be $2({\cdot}6) = 1{\cdot}2$.

"A" can be replaced by "B" once, with probability $0{\cdot}6$. Thus entry m_{12} is $0{\cdot}6$.

B cannot be replaced by any other non-terminal, thus m_{21} and m_{22} both equal 0.

Hence,

$$M = \begin{bmatrix} 1{\cdot}2 & {\cdot}6 \\ 0 & 0 \end{bmatrix}$$

$$M - \lambda I = \begin{bmatrix} 1{\cdot}2 - \lambda & {\cdot}6 \\ 0 & -\lambda \end{bmatrix} .$$

Setting the determinant equal to zero, and evaluating gives:

$$\begin{vmatrix} 1{\cdot}2 - \lambda & {\cdot}6 \\ 0 & -\lambda \end{vmatrix} = 0$$

$$(1{\cdot}2 - \lambda)(-\lambda) = 0$$

$$\lambda = 0 \text{ or } 1{\cdot}2$$

Thus the principal eigenvalue is $1{\cdot}2$, and the grammar is inconsistent.

8.4.3 Defining a working grammar

Quite apart from the problems posed by ensuring that a grammar is consistent, defining a grammar to suit a particular set of conditions is quite a formidable task. The following example shows how a grammar, designed to generate a score for arranging some electronic sounds, was put together.

Example 8.4.3.1

Certain electronic materials were prepared on tape. These were grouped into the following classifications which represent the terminal events for the form generating grammar.

a	static, unchanging sounds, like machines
b	powerful, dynamic, moving sounds, like automobiles or explosive sounds
c	weaker dynamic sounds, like single gun shots
d	sounds suitable for layering in mixtures
e	sounds which must be layered
f	vocal-like sounds
g	events which must occur singly
h	silence

A few simple rules were decided upon:

1. There should be general movement from 'a' through 'b' then 'c' to finish with 'f'.
2. These sections should merge into each other.
3. 'd', 'e', 'g' and 'h' may occur anywhere.
4. 'f' may make occasional appearances near the beginning.

These constraints were then used to define the following grammar.

$$G_{S8} = \langle V_{N8} , V_{T8} , \mathcal{P}_8 , \mathcal{S}_8 , \mathcal{D}_8 \rangle$$

where $V_{N8} = \{A , B , C , D , E , F , N\}$

$V_{T8} = \{a , b , c , d , e , f , g , h\}$

The non-terminals in V_{N8} have particular functions:

A, B and C stand for material which basically consists of a, b, and c respectively, but may be mixed with other types.

D, E and F consist entirely of d's, e's and f's respectively.

N stands for material which can be inserted anywhere in the form structure.

P_8 consists of the following production rules. For convenience the values of $\mathcal{D}_8$ are listed at the side.

S_8	→	A B C F	1	(1)
A	→	A N A	·2	(2)
A	→	A B	·2	(3)
A	→	A F A	·07	(4)
A	→	A / A	·13	(5)
A	→	a	·4	(6)
B	→	B N B	·15	(7)
B	→	A B	·15	(8)
B	→	B C	·15	(9)
B	→	B F B	·05	(10)
B	→	B / B	·1	(11)
B	→	b	·4	(12)
C	→	C N C	·15	(13)
C	→	B C	·15	(14)
C	→	C F C	·15	(15)
C	→	C / C	·15	(16)
C	→	c	·4	(17)
D	→	D / D	·35	(18)
D	→	d	·65	(19)

E	→	E / E	•3	(20)
E	→	e	•7	(21)
F	→	F F	•25	(22)
F	→	F / F	•25	(23)
F	→	f	•5	(24)
N	→	D	•15	(25)
N	→	E / E	•15	(26)
N	→	h g h	•15	(27)
N	→	h	•15	(28)
N	→	∅	•4	(29)

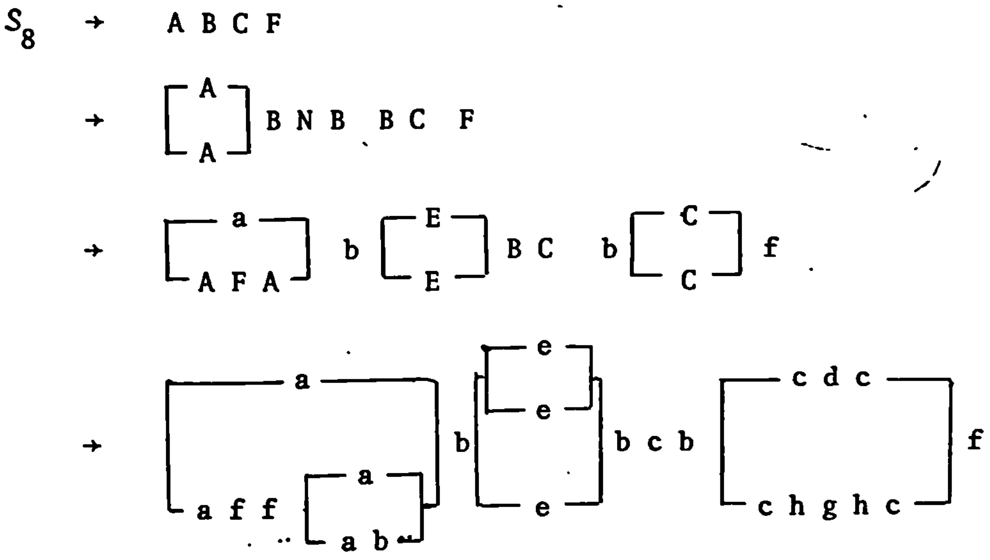
Rule 1, $S_8 \rightarrow$ A B C F specifies the overall form progression.

Rule 26, N → E / E ensures that the 'e' material will always be split into at least two layers.

Rule 27, N → h g h ensures that g is always framed by silence.

Rule 29, N → ∅ (the null production) saves writing more alternatives for rules 2, 7 and 13 as A → A A etc..

An example of the type of generation sequence this grammar could produce is the following:



S8 → A B C F
→ A A B N B B C F
→ a A F A b E E B C b C C f
→ a a f f a a b b e e e b c b c d c c h g h c f

However, when realised in the form of a computer program, and run, the grammar refused to work. Without the need for a full consistency test, simple inspection reveals that the probabilities for a non-terminal becoming a terminal are too low compared with probabilities of it being replaced by other non-terminals. Consider the rewriting rules for non-terminal A, for example. Quite apart from the possibility of being rewritten via another non-terminal, the total of the probabilities for A to be replaced by itself are:

•4 + •2 + •14 + •26 = 1•0 (see rules 2 to 6)

This is much too high compared to the probability of terminating: 0•4 .

It is not an ideal solution simply to adjust the probabilities to make the grammar consistent. This would increase the probabilities of applying the terminating productions for A, B, C and F to about 0•8 , and would in consequence make the possibility of just generating the basic sequence "a b c f" very strong.

One alternative is to count the number of times a non-terminal is replaced by itself, then as soon as a particular limit is reached, all but the terminal production for that non-terminal have their probabilities reduced to zero so that every subsequent replacement will be by a terminal. Such a solution is particularly appropriate for this example where a fixed amount of material in each of the terminal classes was available. This procedure was adopted in the program described in the following section.

8.4.4 A generative Study

In the previous example a grammar was laboriously assembled to define a structure and perform a task which may more easily have been carried out manually. However, the generative scope of stochastic grammars is far wider than suggested by such an example.

8.4.4.1 A splitting grammar defined

A short study was undertaken using the simplest possible web grammar to examine the variety and structure in the patterns generated. The grammar G_{S9} is defined as follows.

$$G_{S9} = \left\langle \{A\}, \{a\}, P_9, A, D_9 \right\rangle$$

where P_9 consists of the following productions:

$$A \rightarrow A\ A \qquad (1)$$
$$A \rightarrow A / A \qquad (2)$$
$$A \rightarrow a \qquad (3)$$

and $D_9 = (p_1, p_2, p_3) \qquad \left(\sum_{i=1}^{3} p_i = 1 \right)$

It can be seen that the grammar has just one non-terminal "A", and one terminal "a".

Following the criteria for consistency developed above, appropriate values of D may be worked out. The first moment 'matrix' is very simple; it has only one entry: $2p_1 + 2p_2$.

Thus, $2(p_1 + p_2) - \lambda = 0$

so that $\lambda = 2(p_1 + p_2)$

Hence to satisfy the consistency criteria and guarantee termination, it is necessary that:

$$2(p_1 + p_2) < 1$$

thus, $p_1 + p_2 < \cdot 5$

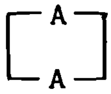
that is, $p_3 > \cdot 5$

This should be apparent simply by examining the original productions and their probabilities.

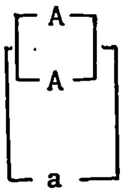
In the string grammars considered earlier, a string of identical terminals would be unlikely to have any apparent structural significance, even though they may have been generated by a complex and sophisticated sequence of productions. However, with a web grammar, the structure itself can be of interest independently of what actual terminals appear. This is the case with the grammar G_{S9}.

A typical generation sequence may be:

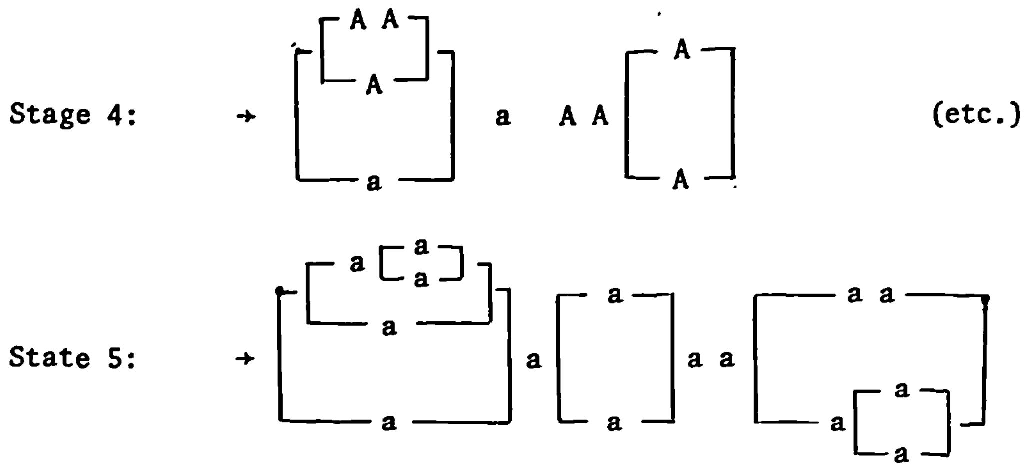
Stage 1: A → A A (rule 1)

Stage 2: → 

A A (rules 2 and 1)

Stage 3: → 

a A A (rules 2,3,3,1)



A useful way of interpreting this sequence of generations is to divide up the two-dimensional plane. Whenever rule 1 is applied, the plane is divided vertically; whenever rule 2 is applied, the plane is divided horizontally; when rule 3 is applied, the dividing process ceases. Thus the above generation sequence would produce the result plotted in figure 97.

Changing the probability assignment $\mathcal{D}_9$ can produce significant variations in the type of pattern produced. If $\mathcal{D}_9 = (\cdot 3\,, \cdot 2\,, \cdot 5)$ then there will be a slight bias towards the first production rule which will produce more vertical versions. Patterns such as those in figure 98 may be produced. The first of these has a very clear vertically split structure. But the second pattern, although constructed from the same probabilities, demonstrates that with such a slight bias, a very even result may be produced.

If $\mathcal{D}_9 = (\cdot 2\,, \cdot 3\,, \cdot 5)$ there will be a slight bias towards the second production rule, which will produce more horizontal divisions. A pattern such as that of figure 99 may result.

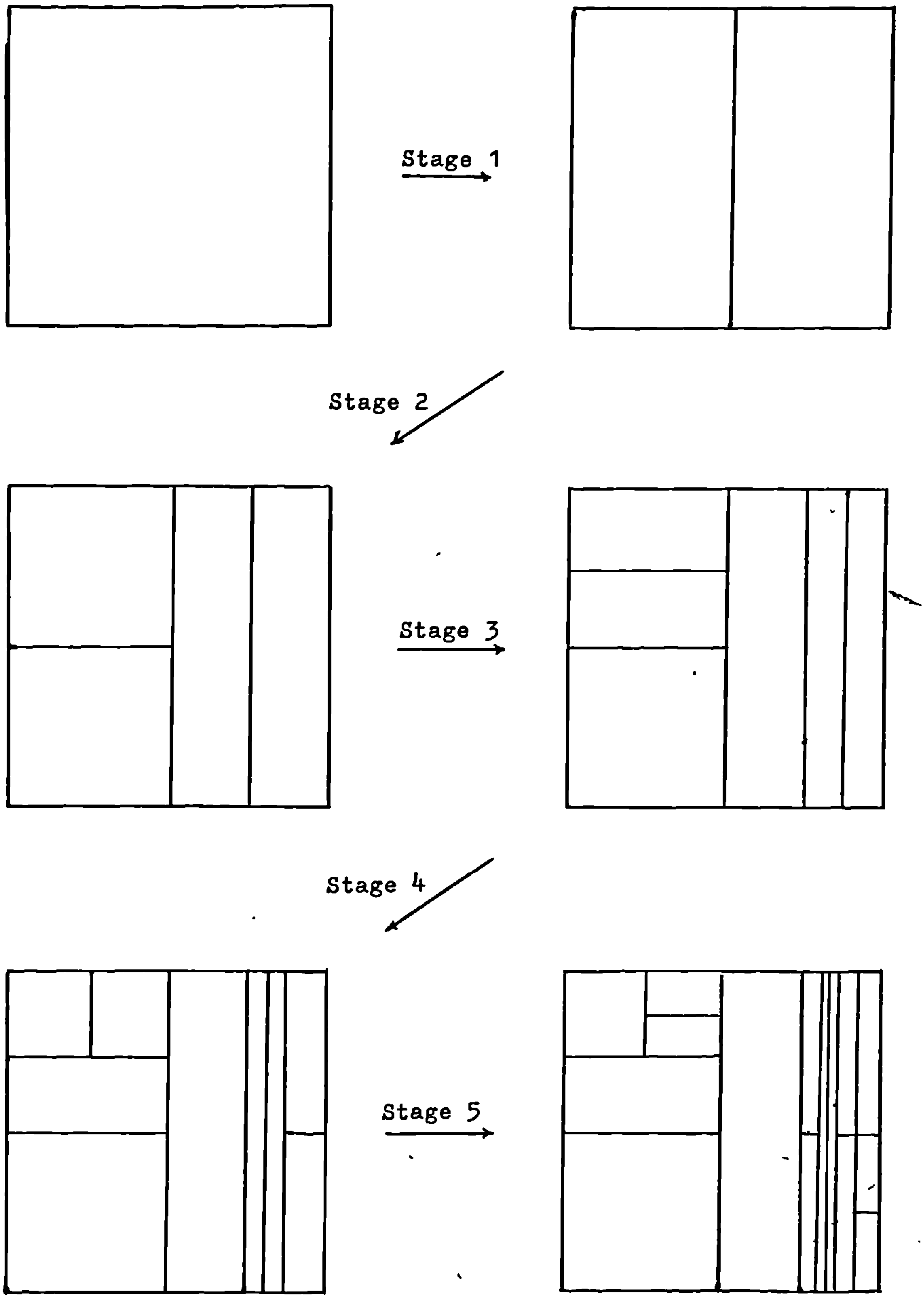


figure 97

Stages in a sequence of generations when applying a stochastic web grammar to divide up the plane

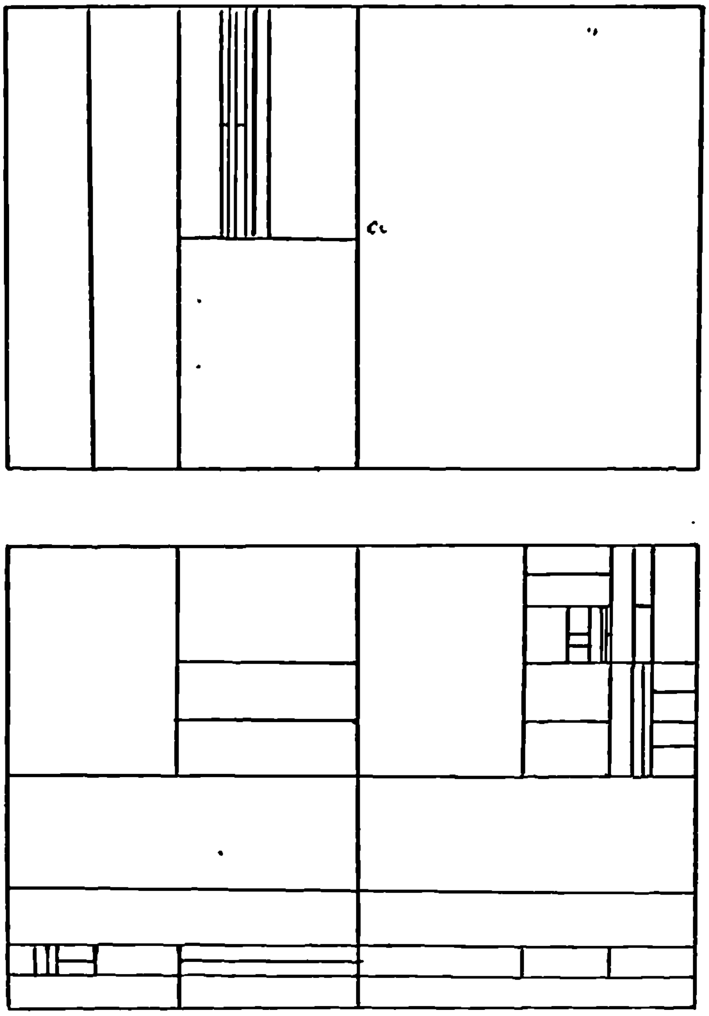


figure 98

Dividing the plane by a simple stochastic web grammar with a slight bias towards vertical divisions

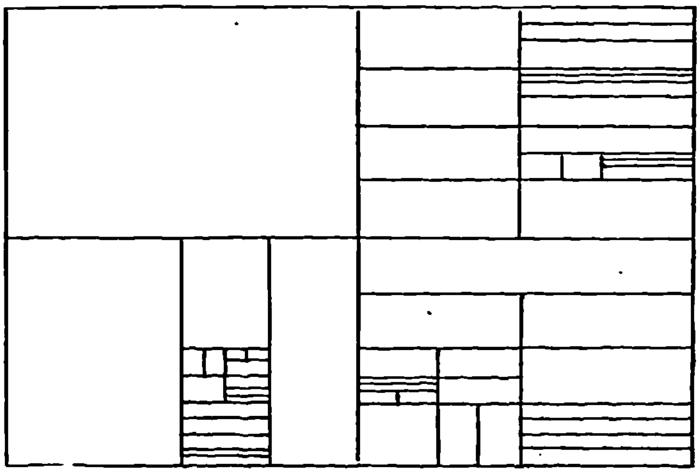


figure 99

Dividing the plane by a simple stochastic web grammar with a slight bias towards horizontal divisions

This same splitting grammar has been used to generate musical structures by dividing up the time/density plane. Whenever a vertical division occurs, the time axis is divided so that two different notes will sound one after the other. Whenever a horizontal division occurs, the density axis is divided so that two different notes will sound together for the same length of time. Thus if $\mathcal{D}_9 = (\cdot 3 , \cdot 2 , \cdot 5)$ as in figure 98, the musical result will be a bias towards *sequential activity* where generally shorter notes will follow one after the other. But if $\mathcal{D}_9 = (\cdot 2 , \cdot 3 , \cdot 5)$ as in figure 99, the musical result will be a bias towards *simultaneous activity* where generally longer notes will sound in continuous chordal groupings.

For the short taped experiment *SPLITS* the p_1 and p_2 probabilities were set equal to each other, which means that both simultaneous and sequential activity are readily apparent. Each note consisted of a simple sine tone, this being the most effective basic unit out of which the complex grammar generated structures could be built. The actual pitches of the notes were determined at random.

By setting just the two values, p_1 and p_2, a wide variety of significantly different structures are obtainable. In using a grammar of this sort, provision is made for structuring both larger scale forms at the macro-level, and for forming distinctive textures and timbres at the micro-level, at one and the same time, as different types of clustered activities are formed depending upon the nature and extent of the divisions of the time/density space.

8.4.4.2 the SPC1 program - for printed output

The ALGOL computer programs used to generate both printed structures and sound structures are very similar with variations in the instructions for scaling and formatting the output.

The most significant part of the program to generate printed structures is the following procedure.

```
      'PROCEDURE' FILL(LH,UH,LV,UV);
         'INTEGER'LH,UH,LV,UV;
         'BEGIN'
            R:=RNDEC(3.5171341791);
            'IF' R > PV+PS 'THEN'
            'BEGIN'
               CT:=CT+1;
               'IF'CT > 2Ø'THEN''GOTO'SKIP;
               FILL(LH,(LH+UH)/2,LV, UV);
               FILL((LH+UH)/2+1,UH,LV,UV);
            'END'
            'ELSE''IF'R > PS'THEN'
            'BEGIN'
               CT:=CT+1;
               'IF'CT > 2Ø'THEN''GOTO'SKIP;
               FILL(LH,UH,LV,(LV+UV)/2);
               FILL(LH,UH,(LV+UV)/2+1,UV);
            'END'
            'ELSE'
SKIP:       'BEGIN'
               L:=ENTIER(RNDEC(9.3171833)*11+1);
               'FOR'I:=LH'STEP'1'UNTIL'UH'DO'
               'FOR'J:=LV'STEP'1'UNTIL'UV'DO'
               RECT[I,J]:=L;
            'END';
         'END';
```

This procedure fills the space specified by the lower and upper horizontal limits (LH and UH) and the lower and upper vertical limits (LV and UV). Making use of the probabilities PH of a horizontal split, PV of a vertical split and PS of terminating, the procedure decides to fill the space either by splitting it in two horizontally and then calling itself again:

```
FILL(LH,(LH+UH)/2,LV,UV);
FILL((LH+UH)/2+1,UH,LV,UV);
```

or by splitting the space vertically and calling itself for both sections:

```
FILL(LH,UH,LV,(LV+UV)/2);
FILL(LH,UH,(LV+UV)/2+1,UV);
```

or by terminating the splitting.

The actual structure was stored in a matrix whose dimensions were read in as data, up to a maximum of 120 × 60 . When a generation was terminated, the part of the matrix defined by the LH, UH, LV and UV limits was filled with the value of an integer, chosen at random, between 1 and 11 . These values represented a scale of 'grey values' which would subsequently be printed out to represent the structure.

To prevent the space from being split too deeply, the detail of which could not be represented by the limits of the matrix dimensions, an additional check was included which counted the number of generations. When the generation depth exceeded 20, the procedure would terminate that particular production.

The remainder of the program, which may be found in appendix E1, deals with reading in the scaled character sequence, matrix dimensions and production probabilities. The generation process itself is effected by the simple call of the procedure:

```
FILL(1,WIDTH,1,DEPTH);
```

And finally, the completed matrix is printed out as a rectangle with the required shading characters.

Two typical results are reproduced in figure 100, for a 72 × 19 matrix, where PV = PH = ·25 .

8.4.4.3 the SPLIT program - for sound data output

The core of the sound generating program is the procedure "A"::

```
           'PROCEDURE'A(START,FINISH,INTENSITY);
              'REAL'START,FINISH,INTENSITY;
              'BEGIN'
                 'SWITCH'TYPE:=MS,HS,SND,SIL;
                 'GOTO'TYPE[RND(4)];
MS:              A(START,START+(FINISH-START)/2,INTENSITY);
                 A(START+(FINISH-START)/2,FINISH,INTENSITY);
                 'GOTO'SIL;
HS:              A(START,FINISH,INTENSITY/2);
                 A(START,FINISH,INTENSITY/2);
                 'GOTO'SIL;
SND:             NOTE(START,FINISH,INTENSITY);
SIL:
              'END';
```

```
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                           ........
??????????----------$$$$$$$$$$$$$$$$$$                  ::::::::::::££...
??????????----------$$$$$$$$$$$$$$$$$$                  ::::::::::::££...
??????????----------$$$$$$$$$$$$$$$$$$                  ::::::::::::££...
??????????----------$$$$$$$$$$$$$$$$$$                  ::::::::::::££...
??????????----------$$$$$$$$$$$$$$$$$$                  ::::::::::::££...
??????????----------$$$$$$$$$$$$$$$$$$                  £££££££££:::££...
??????????----------$$$$$$$$$$$$$$$$$$                  £££££££££:::££...
??????????----------$$$$$$$$$$$$$$$$$$                  £££££££££:::££...
??????????----------$$$$$$$$$$$$$$$$$$                  £££££££££:::££...
```

```
.....................................%%%%%%%%%%?????%%%?????????????????
.....................................%%%%%%%%%%?????%%%?????????????????
.....................................%%%%%%%%%%?????%%%?????????????????
.....................................%%%%%%%%%%?????%%%?????????????????
.....................................%%%%%%%%%%!!!!!%%%?????????????????
.....................................%%%%%%%%%%!!!!!%%%?????????????????
.....................................!!!!!!!!!!...++%%%?????????????????
.....................................!!!!!!!!!!...++%%%?????????????????
.....................................!!!!!!!!!!...++%%%?????????????????
..................................................++%%%?????????????????
.....................................!!!!!!!!!!!!!!!!!!?????????????????
.....................................!!!!!!!!!!!!!!!!!!?????????????????
.....................................!!!!!!!!!!!!!!!!!!?????????????????
.....................................!!!!!!!!!!!!!!!!!!?????????????????
.....................................!!!!!!!!!!!!!!!!!!?????????????????
.....................................++++++++++++++++++?????????????????
.....................................++++++++++++++++++?????????????????
.....................................++++++++++++++++++?????????????????
.....................................                  ?????????????????
```

figure 100

Computer generated structures from the SPC1 program

This procedure produces material from START to FINISH which adds up to a specified maximum intensity. When the first production rule, for a vertical split in the time domain, is applied, the procedure calls itself again twice; once for the first half and again for the second. When the second rule, for a horizontal split in the density domain, is applied, the procedure again calls itself twice, but with the same start and finish limits. This time, however, the intensity is halved. This ensures that the maximum value permitted for the sound synthesis program input data is not exceeded.

For additional variety in the resulting sound output an extra terminal production possibility has been added. Either a note is defined, of random pitch and with time and intensity specifications derived from the procedure parameters applying when it was called, or alternatively, no action is taken resulting in silence for the start to finish limits. This is essentially equivalent to defining a variant on the original grammar of the form:

A	→	A A	·25	(1)
A	→	A / A	·25	(2)
A	→	a	·25	(3)
A	→	∅	·25	(4)

where the fourth rule is the *null* production which merely reserves a space in the structure without filling it with anything.

In this program, the four options are given equal probabilities of occurence.

When an actual note is to be defined, the precedure ''NOTE(P,Q,R)' is called. (See appendix E2 for complete program listing). This transfers the note parameters into the correct format for MUSIC 5 data input producing a series of note definitions each of the following form:

```
  = note            end time        pitch (Hz)
    ↓                  ↓                ↓
   NOT     Ø     1     1Ø     187.5     2747     Ø     Ø;
           ↑                    ↑
       start time           intensity
```

The complete list of note data produced for the tape example may be found in appendix E3. This note data was input to the MUSIC 5 sound synthesis program and run on the ICL 1900 computer at the City University, London.

A pitch/time score, derived from this data is given in figure 101 (pages 273 - 5). Intensities have been grouped into just four classes which correspond to the values in the data list as follows:

pp	0 - 0·9999
p	1 - 9·9999
f	10 - 99·9999
ff	100 - 1 500

It should be pointed out that due to the successive halving of intensities in the original SPLIT structuring program, some notes end up being so soft that they are inaudible against other notes sounding simultaneously with a greater intensity.

figure 101

A pitch/time score of sound generated by the SPLT program using a stochastic grammar

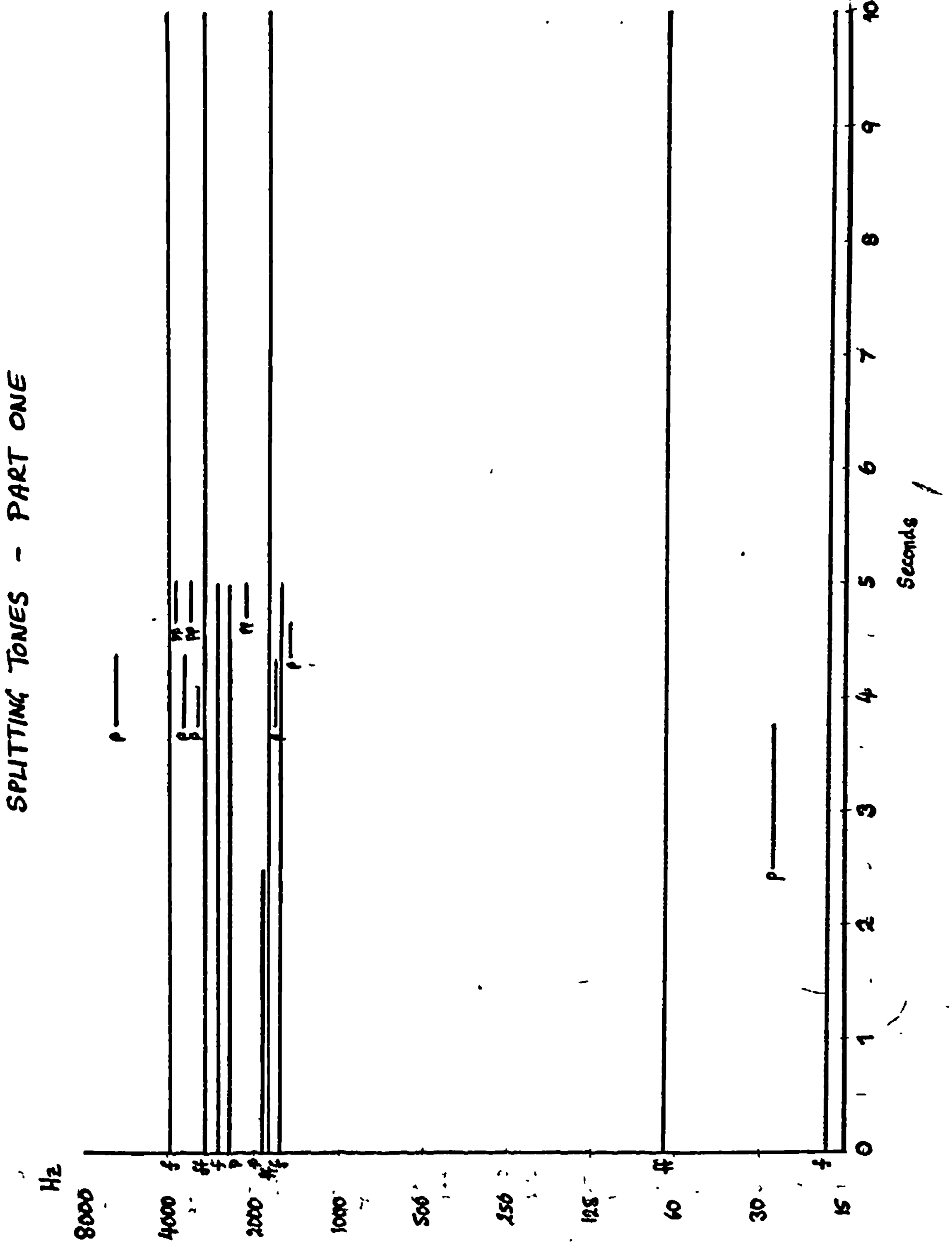


figure 101 (continued)

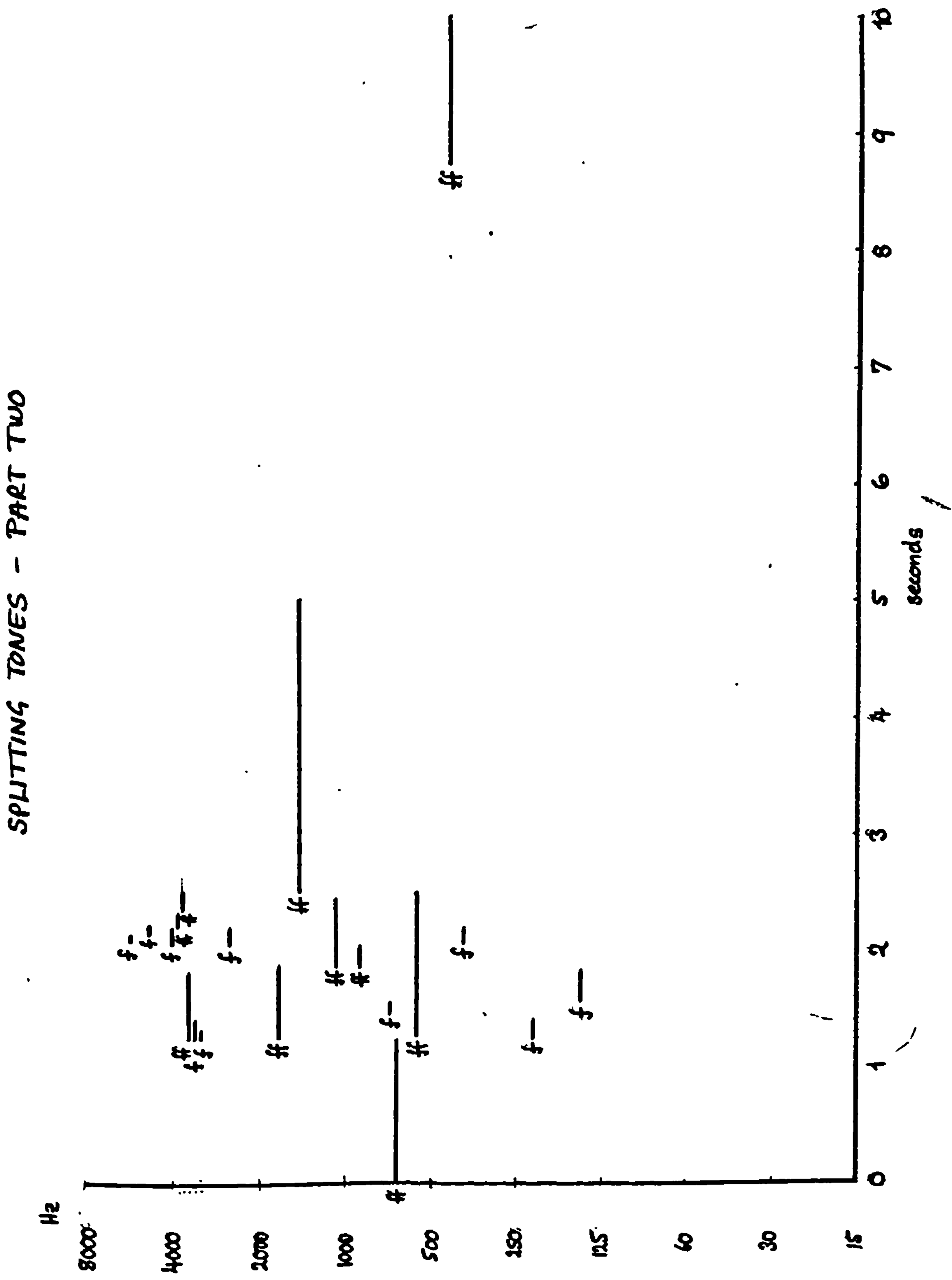
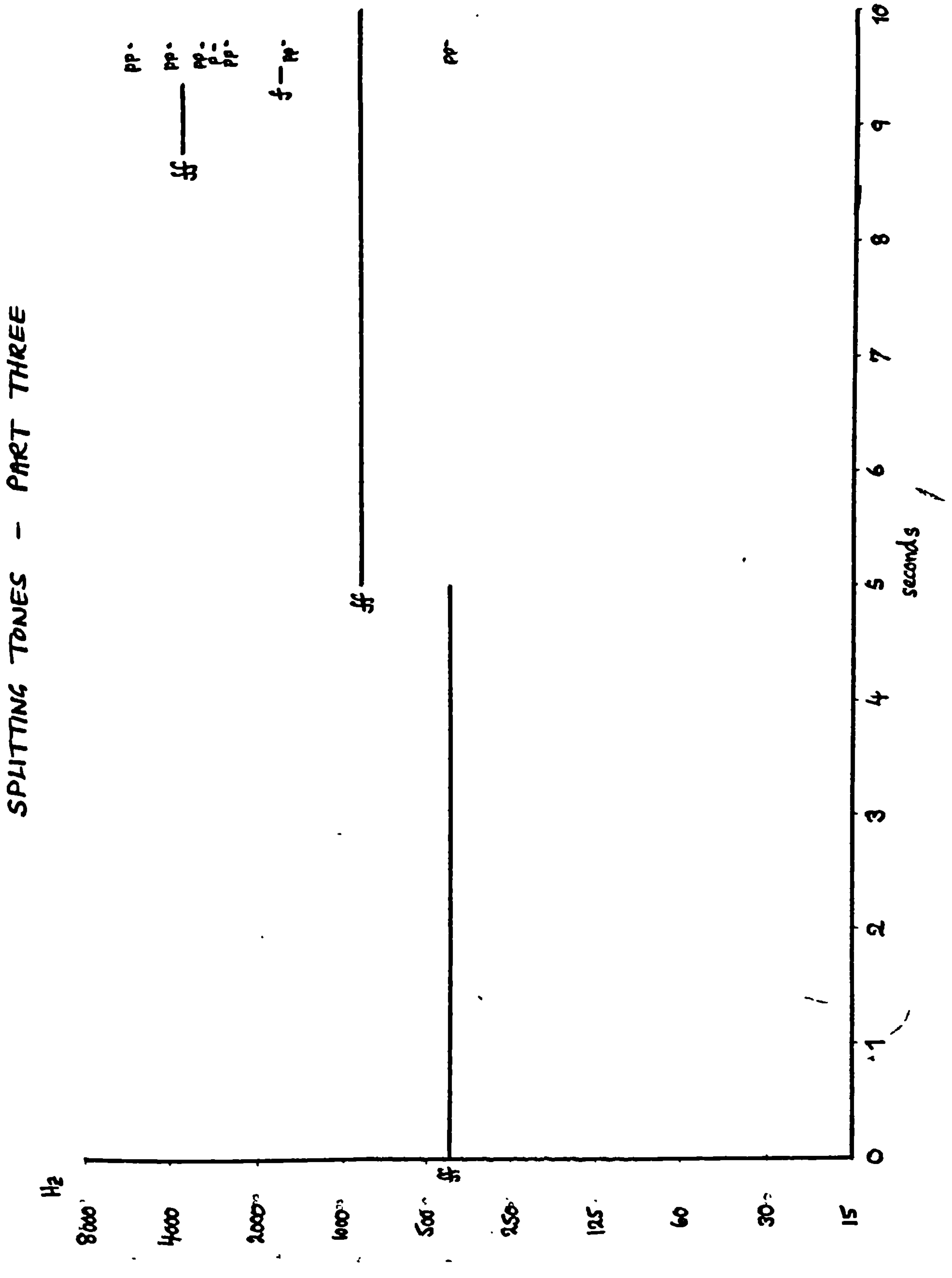


figure 101 (continued)



8.4.5 Extension into more dimensions

The space splitting grammar concept can be extended into three or more dimensions. The following grammar can be used to split up a block in three dimensions (like cutting up a block of butter). The indexed operator, $/_n$, will be used, indicating a split in the n-th dimension.

$$G_{10} = \left\langle \{A\}, \{a\}, P_{10}, A, \mathcal{D}_{10} \right\rangle$$

where P_{10} consists of the following productions, with associated probabilities for $\mathcal{D}_{10}$.

$$\begin{array}{lllll} A & \rightarrow & A /_1 A & p_1 & (1) \\ A & \rightarrow & A /_2 A & p_2 & (2) \\ A & \rightarrow & A /_3 A & p_3 & (3) \\ A & \rightarrow & a & p_4 & (4) \\ A & \rightarrow & \emptyset & p_5 & (5) \end{array}$$

$$\sum_{i=1}^{5} p_i = 1$$

and for consistency, $p_1 + p_2 + p_3 < \cdot 5$

$$p_4 + p_5 > \cdot 5$$

The following is a typical sequence generated by this grammar, where,

$$\mathcal{D} = (\cdot 15, \cdot 15, \cdot 2, \cdot 25, \cdot 25)$$

$$(\emptyset /_3 ((\emptyset /_2 a) /_3 a)) /_3 ((a /_1 \emptyset) /_1 (a /_2 \emptyset))$$

This sequence, in fact, corresponds to the three-dimensional web-diagram given at the end of section 8.3 (figure 95). It can be represented by slicing a three-dimensional block in half, width-ways when rule 1 is applied, in height when rule 2 is applied, and length-ways when rule 3 is applied. This is shown in figure 102(a), and is perhaps clearer when the chunks are separated, as in figure 102(b). Chunks terminated by rule 4 are shaded, chunks generated by rule 5, the null production, are not. These empty chunks have been removed in figure 102(c) leaving only the block which has been generated by the grammar.

Two additional block structures generated by this grammar are given in figures 103(a) and (b) (page 279). Each block in this diagram is the inverse of the other, where a's have been replaced by Ø's and vice versa.

Varying the relation between the p_4 and p_5 probabilities varies the density,of the structure produced. If p_5 is very small, a cheese-like structure will result containing a few holes. If p_5 is large, a few blocks will be left suspended in a largely empty space.

When applied in a musical context, musical parameters can be assigned to any dimension as required.

Example 8.4.5.1

The structure in figure 103(a) (page 279) can be interpreted as a musical score by making height equivalent to pitch, width to time and length to timbre. The timbral domain could be defined in terms of a progression of frequency modulation values, or simply as an arbitrary set of instrumental timbres. The block structure can be rewritten in a form more closely equivalent to a conventional score by taking a series of slices lengthways through the block, one for each timbre. These slices can then be arranged one below

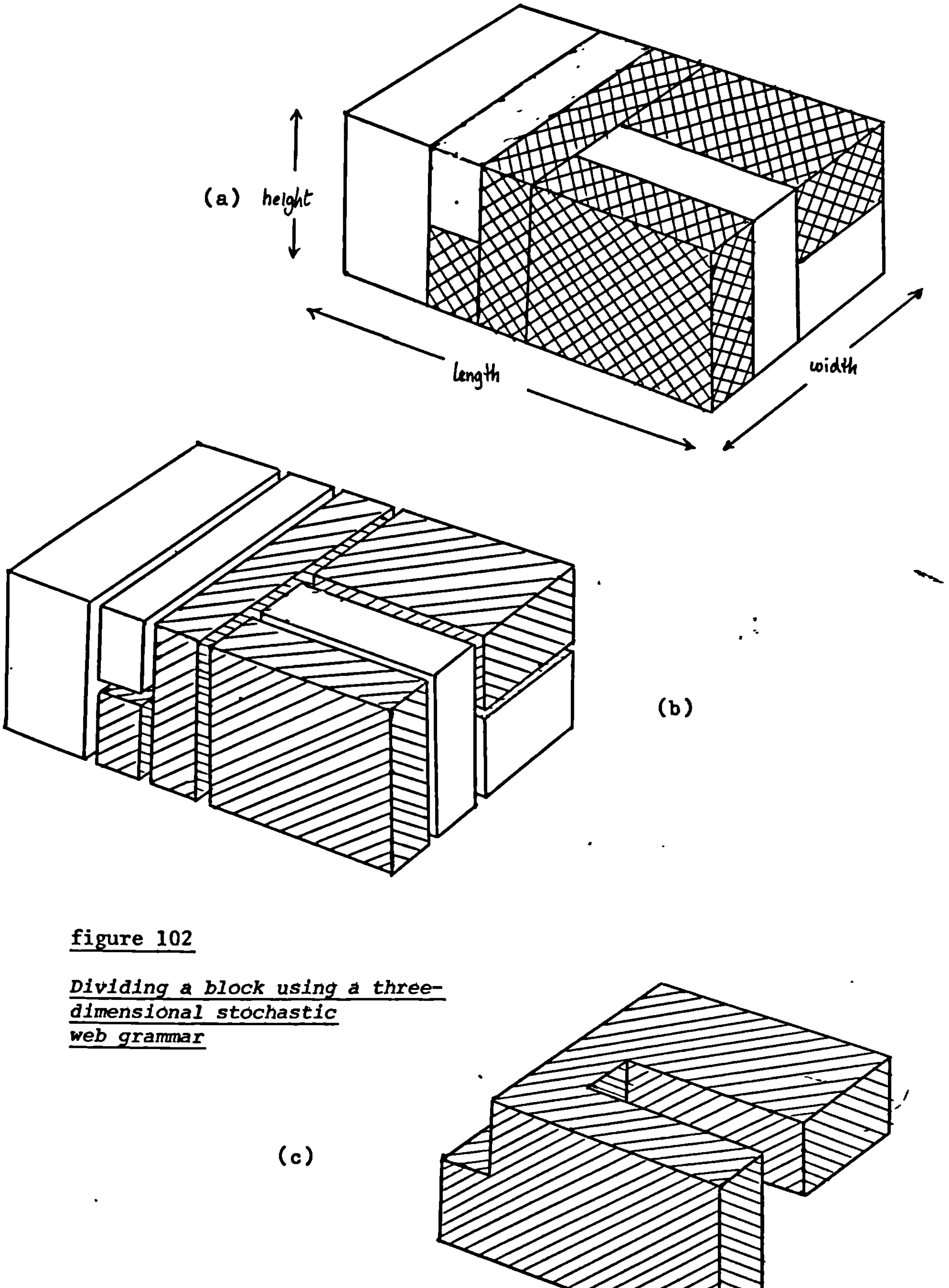


figure 102

Dividing a block using a three-dimensional stochastic web grammar

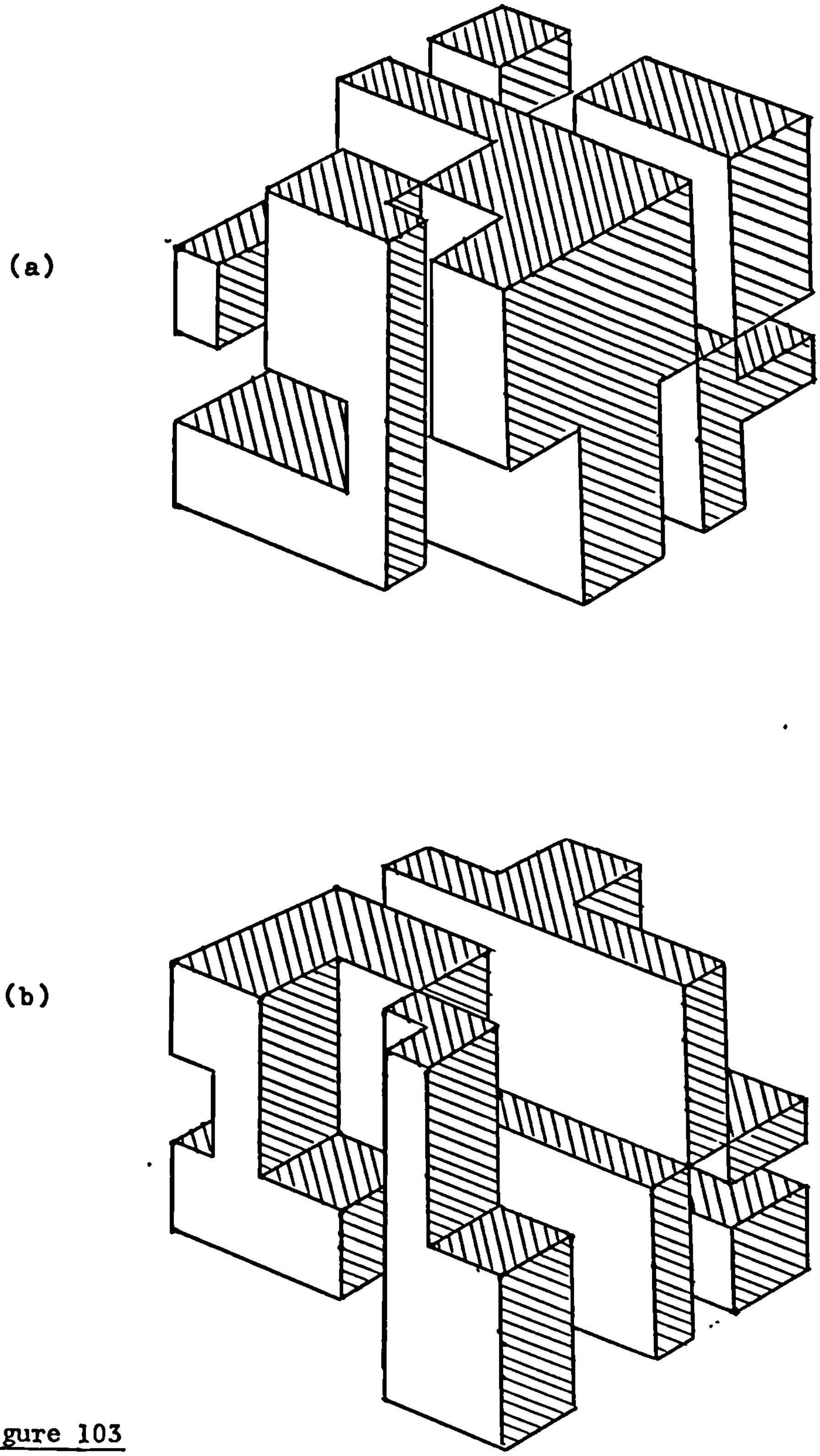


figure 103

A block and its inverse generated from a three-dimensional stochastic web grammar the other, as in figure 104, just as is done in a conventional score.

In the above example the scale of the time domain is not specified. It could represent one bar, a few seconds,a whole page or a whole piece. This applies similarly to the pitch domain where only general areas of pitch are identified. In order not to complicate the example, the number of generations were deliberately limited. However, when they are allowed to continue to a much deeper level , as has already been emphasized above, aspects of both micro-structure and macro-structure are taken care of. Each small element of the structure is directly related to a larger progression of parent structures which stretch across all of the dimensions in use.

Far more sophisticated musical applications in generating compositional structures may be made by building upon the basic principles of a simple splitting grammar in extending and generalising its various aspects. The grammar can be expanded into more dimensions to cover as many parameters as are needed to define sound structures. In addition, larger non-terminal and terminal sets may be defined, and the values in $\mathcal{D}$ can be made to vary during generations according to certain conditions. If the $\mathcal{D}$ probabilities are defined to be dependent upon generation depth, patterns may be constructed with macro- and micro-structures of differing characteristics. For example, a macro-structure may consist of a predominantly vertically split sequence of micro-structures which themselves are split horizontally. Alternatively $\mathcal{D}$ may depend on what generations have already taken place, to force the production of antithetical structures for example, or $\mathcal{D}$ may change quite arbitrarily

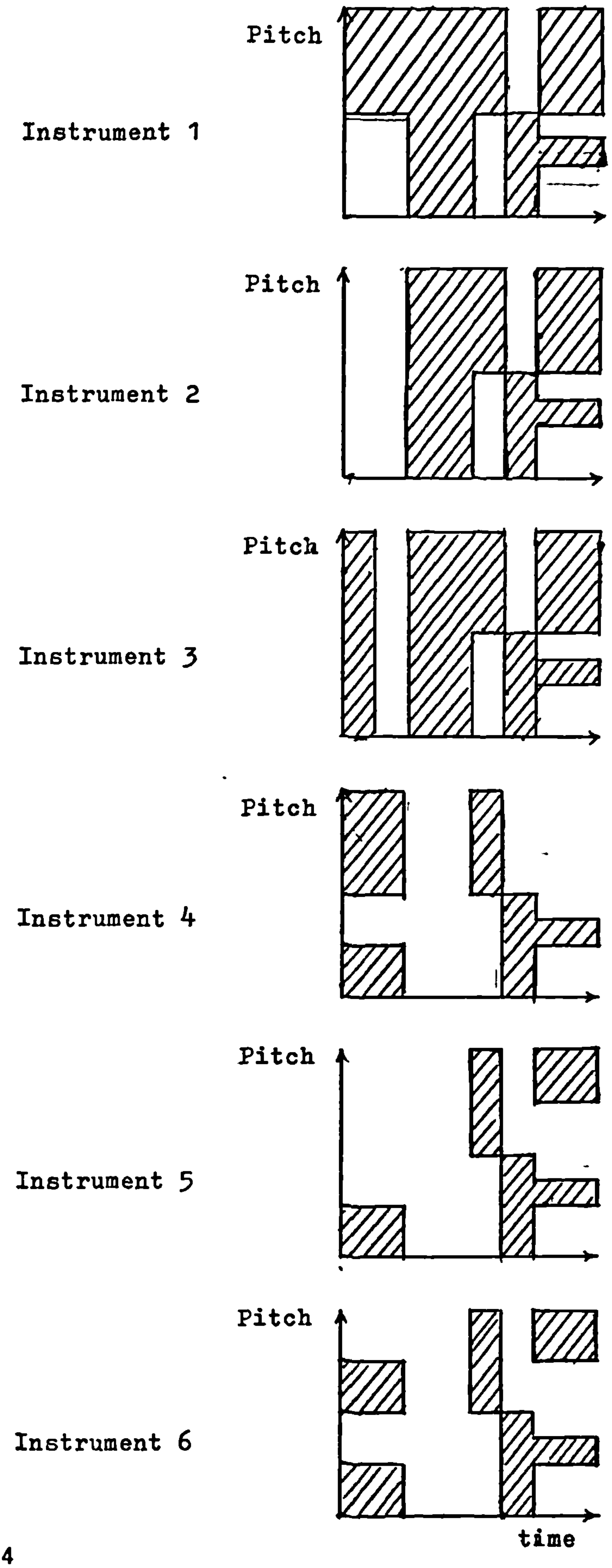


figure 104

A partial musical transcription of the block in figure 103 in the middle of a generation sequence. In this way, solid structures may be formed looking like the cooling fins on a motorbike engine (figure 105). In a musical context this would correspond to a situation where one group of instruments produced a sequence of short, separated block chords whilst another group simultaneously held static lines of longer notes. Such a structure is fairly common in many musical genres. Quite a dramatic example occurs towards the end of the Grand March from Verdi's *Aida*.[10] This example from Italian grand opera is perhaps appropriate for comparison with the Italian motorbike engine (a Moto Morini) in figure 105.

In the above examples the 'split' has always been into two equal parts. An interesting possibility is to make each split at the Golden Section, (roughly in the proportions 1·618 to 1). Repeated application would produce structural divisions which were related by the Fibonacci series.[11] This corresponds to the proportions of natural growth patterns apparent in the structure of most plants, and has influenced the work of many composers from Machaut and Dufay [12] to Bartok.[13]

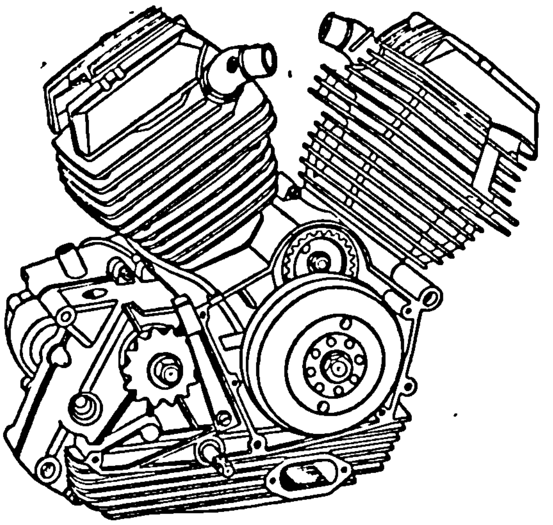


figure 105

Cooling fins on a motorbike engine

8.5 A note on Heuristic or Self-regulating grammars

A brief outline will be given indicating how stochastic grammars may be practically applied in a heuristic context.

A *heuristic* or *self-regulating* grammar is a grammar that is capable of adapting itself to a user's requirements through a feedback response network. It is useful in a computer assisted learning environment where the user is 'protected' from the entrails of the system and merely responds to its promptings.

The most usual way for many composers to work, no matter how experienced they are, is to 'fiddle around' with a system to see what it can do. Unfortunately, the more sophisticated the system, the more preparation is needed, and the greater the possibility of getting nowhere fast. This frequently happens when beginners experiment with a synthesizer, for example. However, within the constraints of an intelligent system, the user can develop ideas using a controlled trial and error process where he responds appropriately to the computer's invitations. After listening to a trial offering from the computer he can reject it, or indicate how close it is to what he requires. The computer can continuously monitor these responses which are used to determine the nature of each subsequent trial. Having homed in on the required sound structure, its pertinent characteristics may be stored for subsequent re-use. This response network is represented in figure 106.

It could be a very difficult task to attempt to define a general generative system which can respond and adapt as described. However, the stochastic

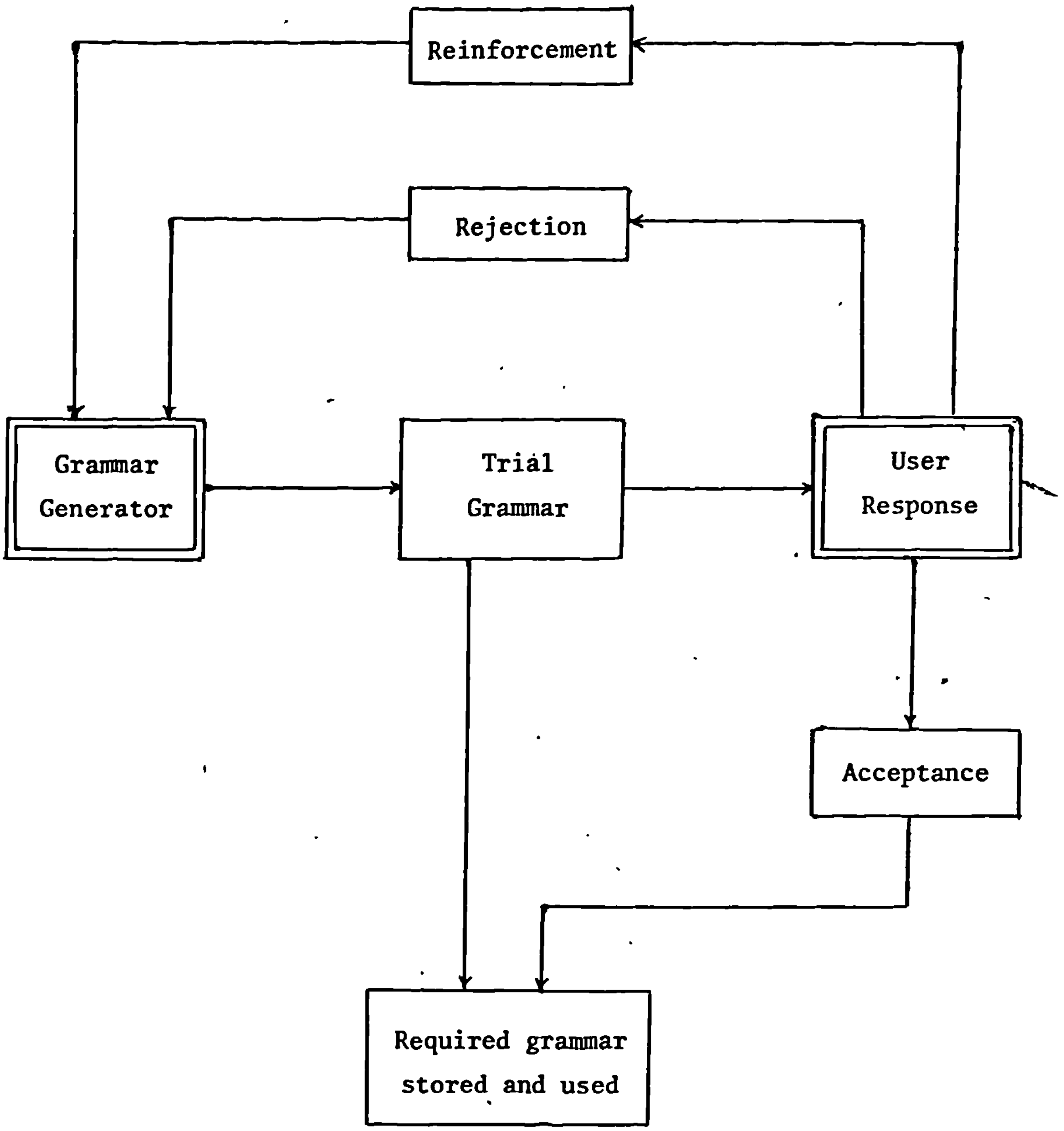


figure 106

Trial and response network for composition with heuristic grammars

grammars under consideration would appear to have significant potential as a useful tool in such a situation.

An original basic *resource grammar* would act as the grammar generator. This would need to consist of a large repertoire of possible production rules, including null productions. When a generated structure was to be reinforced, the production probabilities in $\mathcal{D}$ associated with the production rules responsible for the structure, would be incremented, with appropriate adjustments made to the remaining probabilities. When a structure was rejected, which is likely to happen most of the time, the production probabilities would be decremented. In this way certain probabilities would eventually become zero making their corresponding production rules redundant. When a particular grammar is eventually acceptable, it can be saved by storing the current values in the $\mathcal{D}$ array which define it. This grammar can then be recalled when required and will always generate structures of a particular character. The saved grammars created in this way would be capable of generating both textures and larger scale forms which may be quite specific and restricted in style, or more general and unpredictable depending upon the way in which the user manipulated his responses.

CHAPTER NINE

APPLICATIONS AND EXTENSIONS

9.1 A summary of the uses of stochastic generative systems

This thesis has presented an expansive theory of stochastic processes in computer music. As the theory has been developed reference has been made to a number of different applications. Even though the theory is presented as an intrinsically integrated study of form and structure in own right, it is none the less appropriate at this stage to summarize and assess the wider range of possible applications.

At the very beginning of the thesis, reference was made to the powerful possibilities of the application of stochastic techniques for data reduction with computer music sound synthesis systems. Their suitability for defining and controlling certain types of structures has been convincingly demonstrated where the techniques have been easily applied to control large massed structures consisting of many tiny elements. This is particularly apparent in the composition *MACRICISUM* described in chapter seven. The techniques are more vitally extended to the powerful self-generative structures described in the last chapter where by defining a basic generative 'womb' it becomes possible to form an infinite variety of significantly different patterns in any number of dimensions simply by varying a few probability values.

It has also been pointed out that stochastic techniques can produce completely unexpected, yet welcome, surprises. In this way it is possible to break out of the bonds of limited imagination. The author has been delighted by the gifts of stochastic serendipity on a number of occasions. Of course, the reverse also applies in that disastrous or useless results may sometimes be generated, but these can easily be rejected. The appearance of just one unanticipated gem amply redresses the balance. When a particularly satisfying result appears, the structures and values responsible may then be exploited to further advantage. When working on sections of *MACRICISUM*, the author was on occasions uncertain about what type of aural result would be generated from a particular structure. This was the case with the movement *LAITRAPARTIAL*, for example, where an interesting result was anticipated, but the richness of what the computer actually produced came as a pleasant surprise.

The heuristic systems described at the end of the last chapter are essentially a formalisation of this process of experiment, discovery and exploitation. Instead of a competent programmer having to plough through the necessary operations, the system described could perform the tasks automatically. Such heuristic systems would appear to have great potential in widening the general application of computer music systems, to become more attractive to ordinary musicians, or indeed to any member of society who wishes to enjoy making music.

9.2 Possibilities for expansion and further research

There are many points raised in the course of the thesis which have had to be left open-ended in order to pursue the main line of enquiry without too much digression. Many of these point have significant potential for further development, and a summary of these is given by way of conclusion.

The difference distributions introduced at the end of chapter four and briefly taken up again in section 6.4.2 would appear to repay more detailed investigation. The possibility of using a difference distribution to construct another distribution, which may be continued through a series of generations, is suggested. Also, the relation to Markov Chains may be considered in greater depth.

The Markov chain systems themselves have much potential for further investigation. The application of simple Markov Chains for certain types of musical analysis would appear to be a rewarding study. The further analysis and application of cyclic Markov Chains (section 6.4.2) should be particularly fruitful in this context. In addition, the extension of Markov Chains using time-variant probabilities, and also in considering the relationship between the component parts of patterns formed by Markov Chains with event vectors (inference in Markov Chains) should both yield useful results applicable in many types of musical analysis and synthesis.

Different forms of stochastic grammars may be constructed and studied, making use of depth-indexing and cross-referencing of the type suggested in section 8.4.5 . And it is just a short step to design a practical composing system based on the ideas outlined in section 8.5 .

Very little work has been done analysing the effect of stochastically defined structures on the listener. Much useful psycho-acoustic research could be done on this aspect of musical perception, to determine what sorts of structures are most clearly distinguishable from each other, and where parameter limits may most naturally be drawn.

Finally, it has been pointed out at various points in the thesis how certain structures resemble other processes in the natural world. The continued development of such resemblances is a fascinating area of study which can be applied practically in the preparation of multi-media performances as well as having wider potential repercussions.

Xenakis, in developing his own theory, claimed that his music followed the same laws as existed in nature; the laws defining the sound of cicadas, the collision of hail with a hard surface, or the corporate sound of a large mass of demonstrators.

> "This mass event is articulated and forms a plastic mould of time, which itself follows aleatory and stochastic laws." [1]

Such a claim, however, is rather extravagant. It is not appropriate to speak of natural phenomena as following stochastic 'laws', which begs certain ontological questions. It is more meaningful to speak of *models* of the real world. A certain stochastic model may be just one of a number of interpretations of a physical phenomenon, each of which may claim equal status. The stochastic interpretation may then be used as a musical template which may be genuinely claimed to model the original phenomenon. Such is the claim that would be made for the musical structures, and their visual and physical counterparts, which are identified in the thesis. Any claim that a stochastic model represents the *actual* forces involved in any phenomena is misleading (in so much as reality can be understood anyway!).

Leibniz pointed out long ago that any finite set of observations (and any physical system can only be subject to a finite set of observations) can be accommodated within an indefinitely large number of explanations.[2] Popper, in his celebrated lecture "Of clouds and clocks", the inspiration of Ligeti's composition with a similar title, identified the two commonly accepted poles of scientific description: *clocks*, which are totally reliable and predictable, the *ne plus ultra* of deterministic activity, and *clouds*, which are unpredictable and describable only in general stochastic terms.[3] He then carefully proceeds to destroy this illusive polarity by demonstrating that all clocks are in fact clouds, and that all clouds have clock-like qualities! Popper thus concludes that the 'correct' interpretation lies somewhere in between. He suggests a heirarchical system of 'plastic controls'; a system of clouds controlled by clouds. Such a description corresponds very closely to some of the structures described in this thesis.

Perhaps these capricious conceptions of a mere musician may be able to make a small contribution to a greater understanding of man's mind and of the world he inhabits.

non hydra secto corpore firmior

vinci dolentem crevit in Herculem

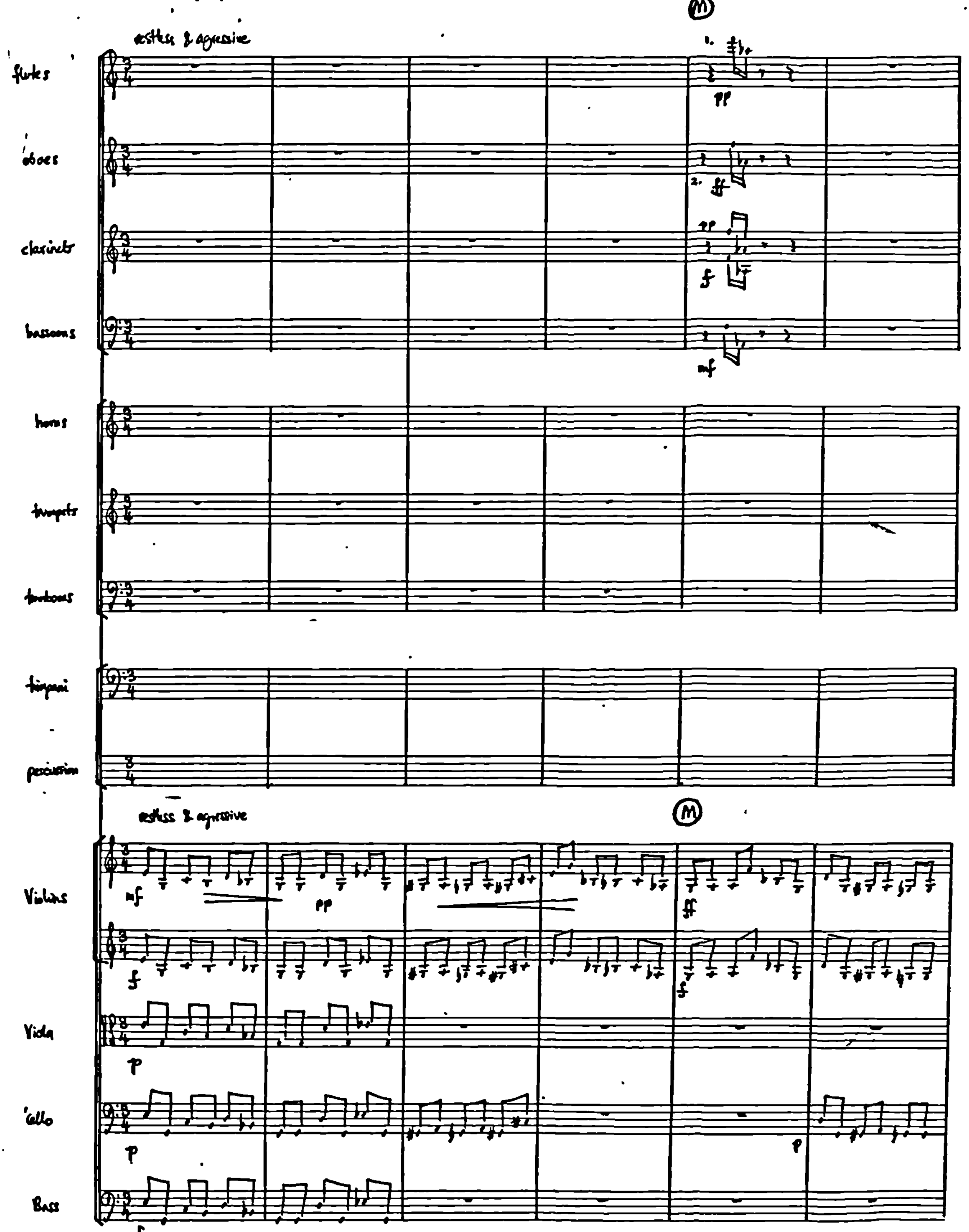
HORACE

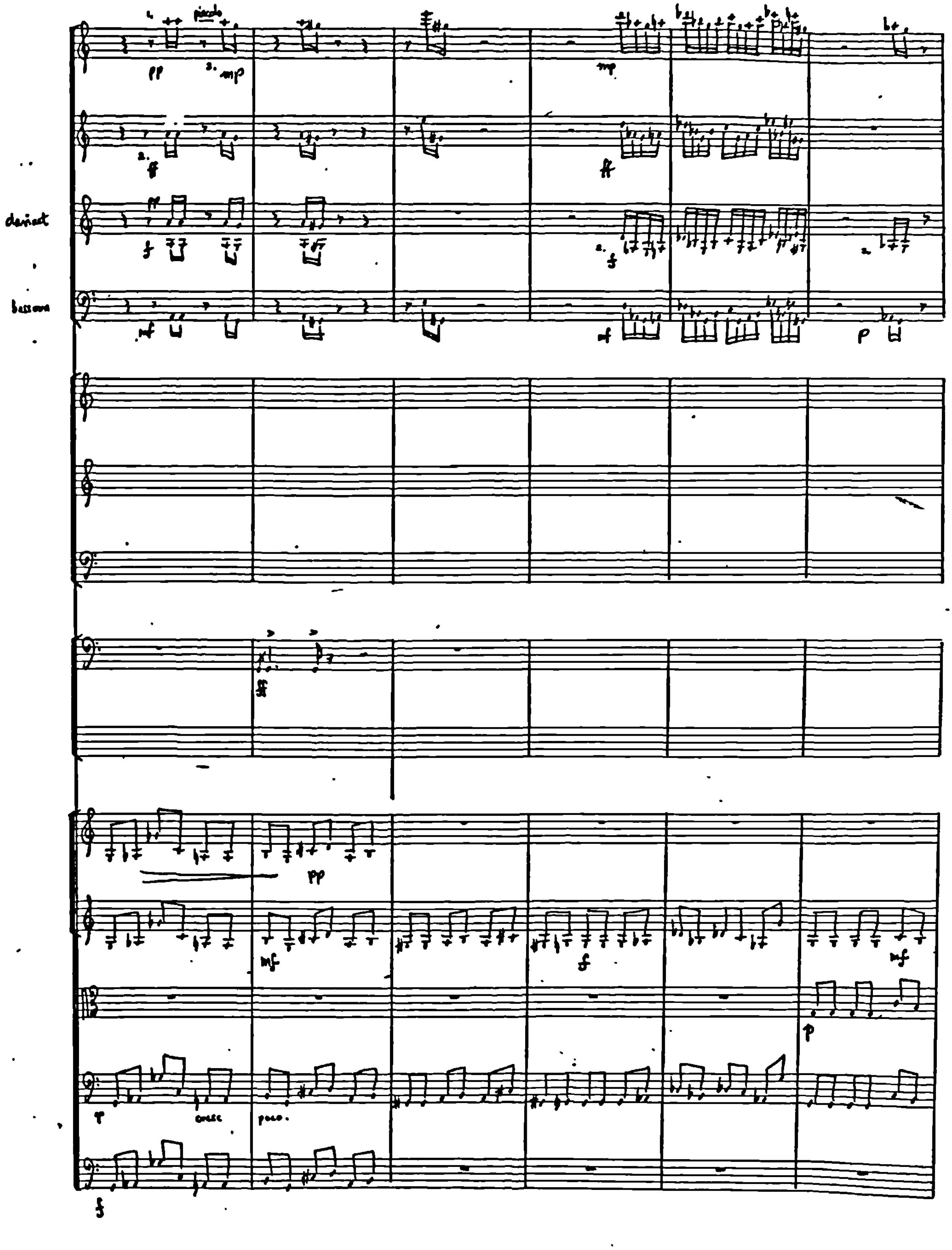
APPENDICES

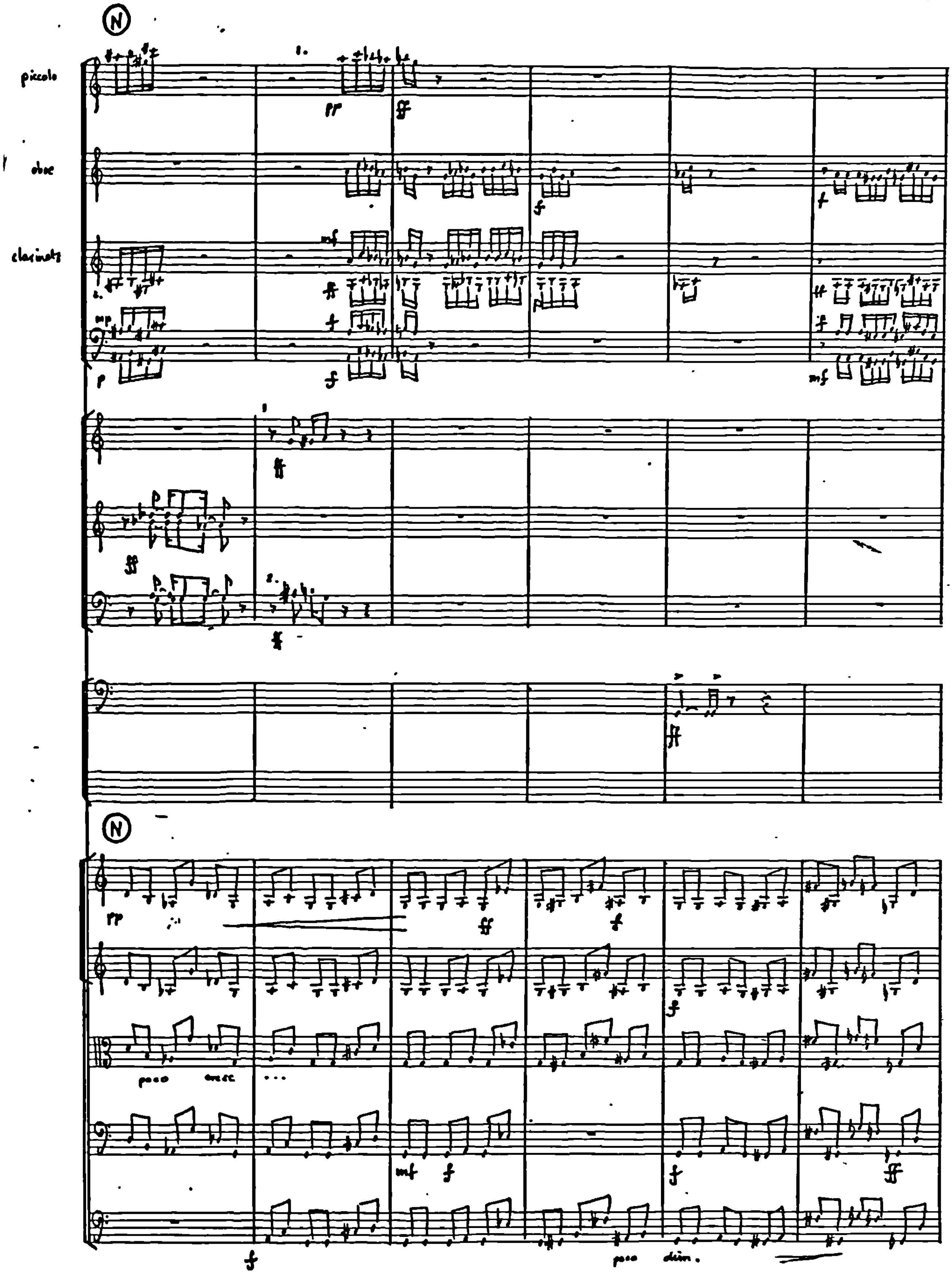
Appendix A

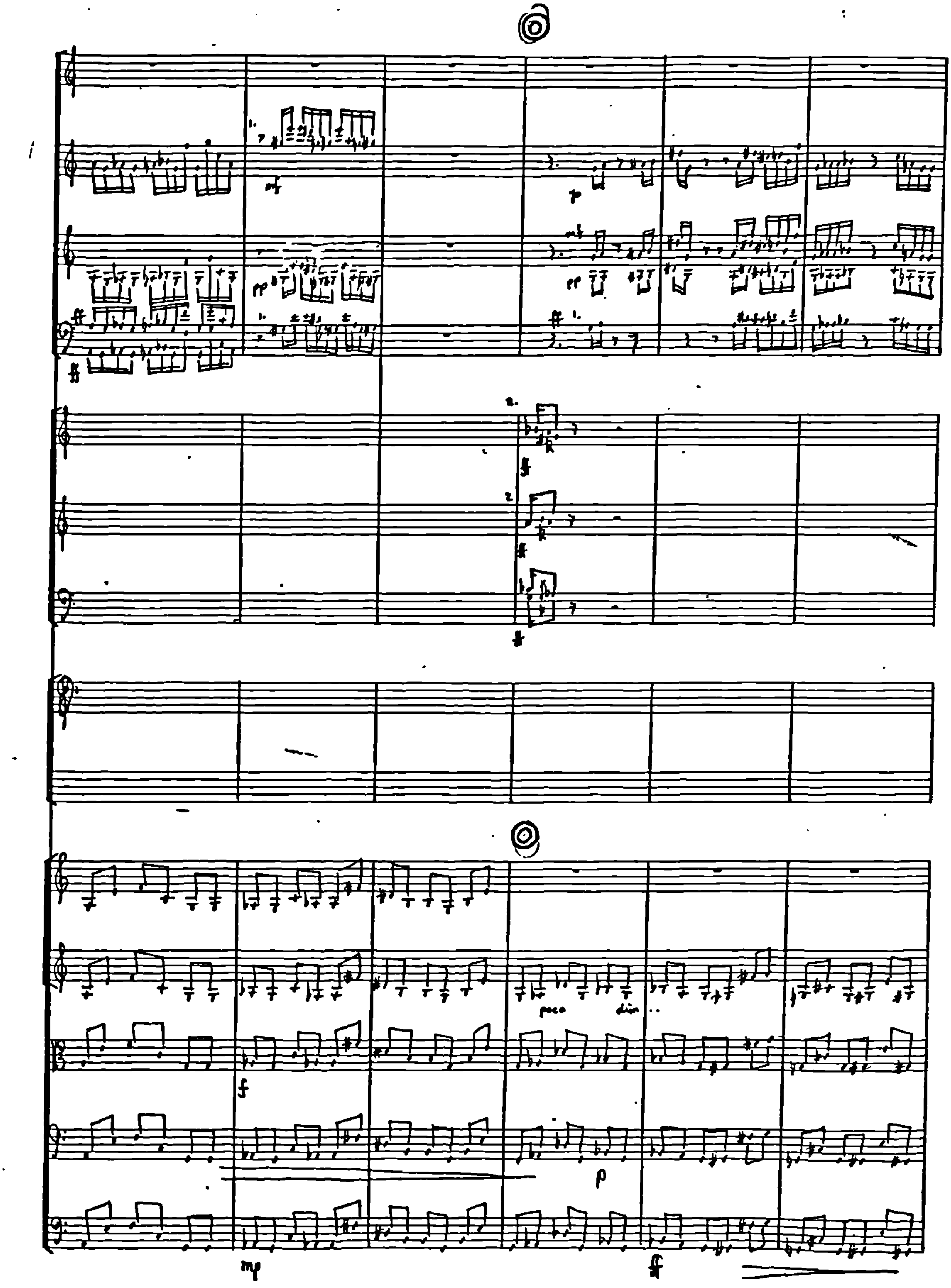
FIRELAKE (Score)

4. lake of fire

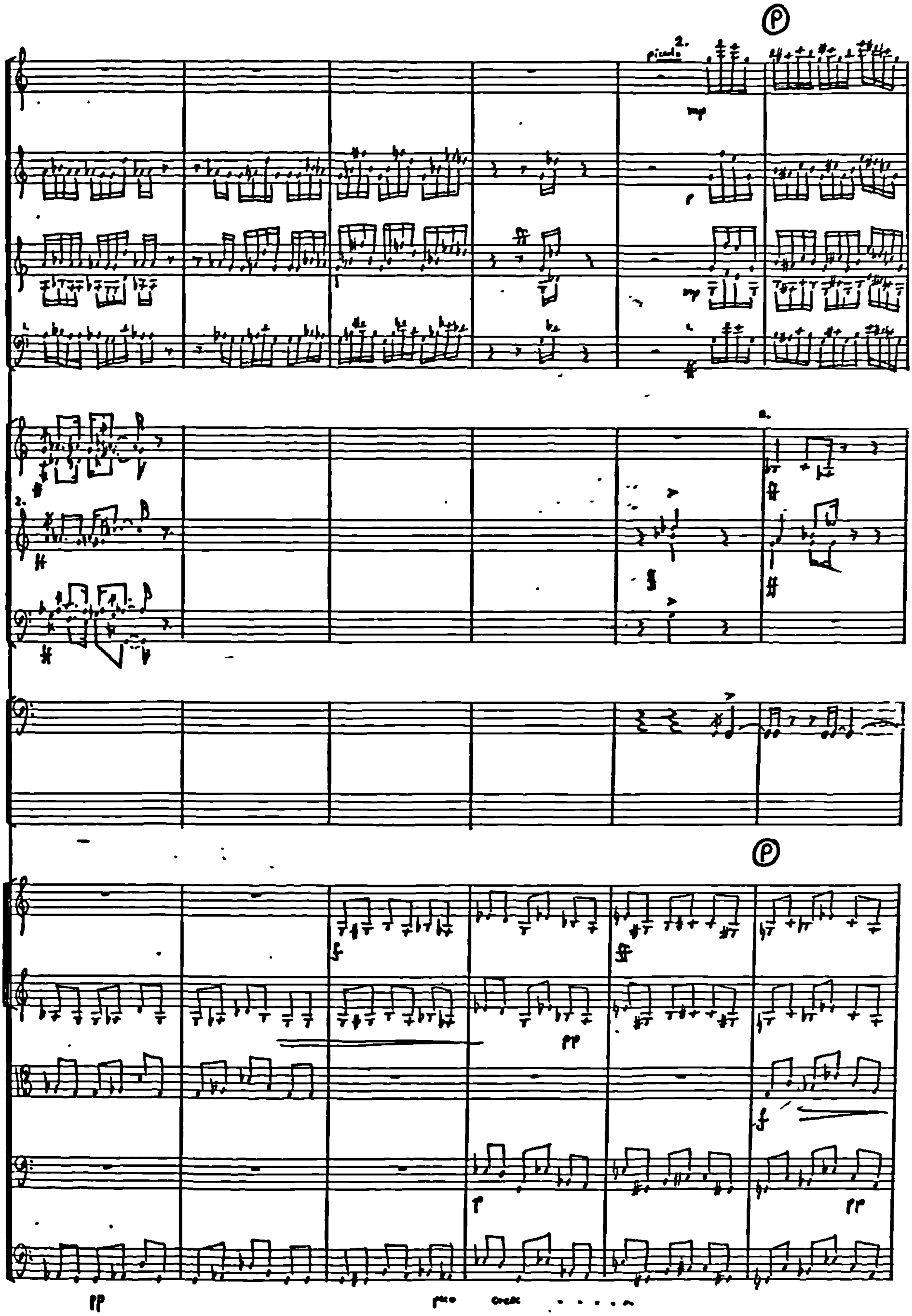


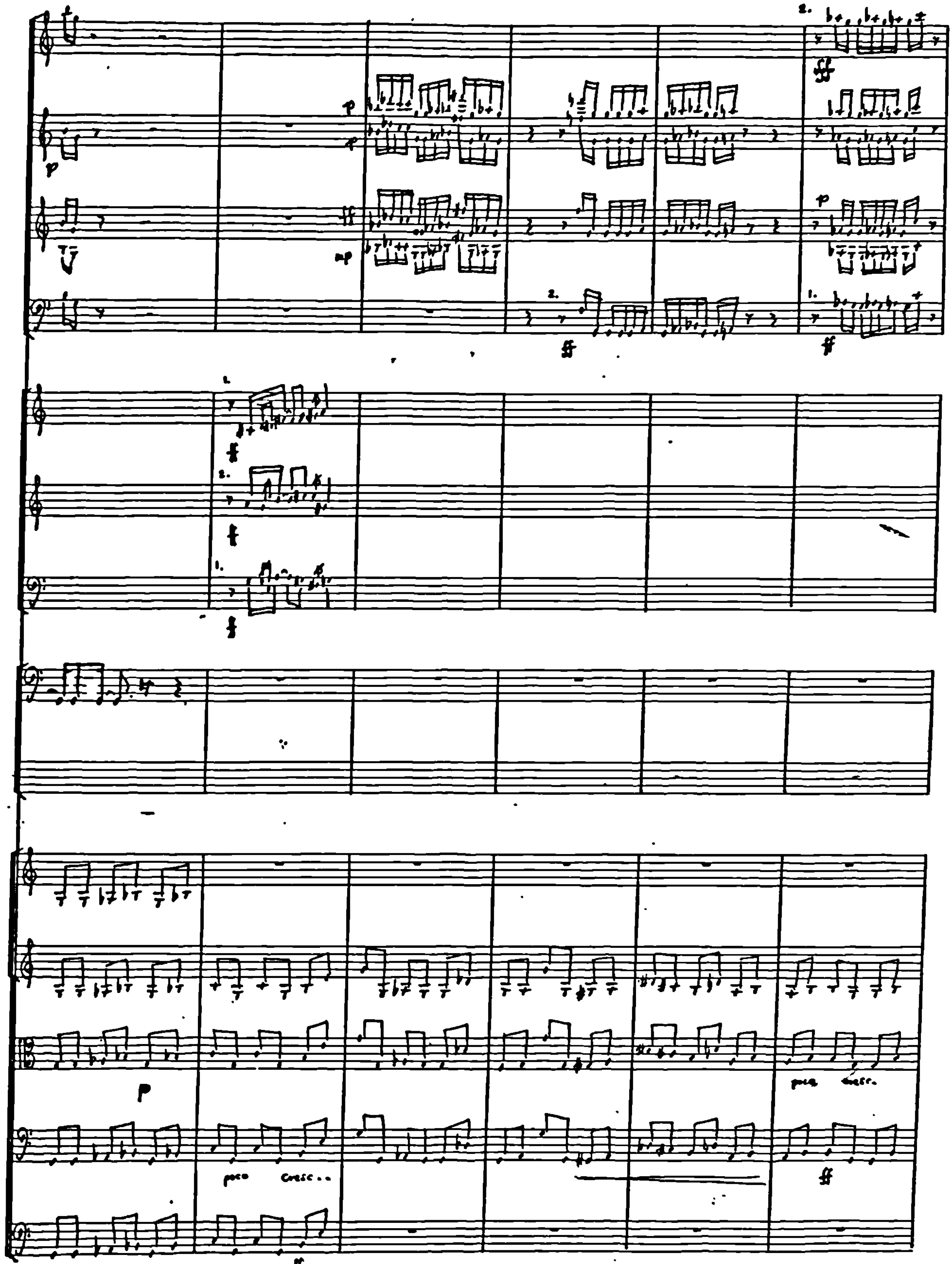

piccolo
pp
2. mp
mp
2. ff
ff
clarinet
f
2. f
2.
bassoon
mf
mf
p
ff
pp
mf
f
mf
p
p
cresc
poco.
f


N
piccolo
oboe
clarinets
pp
ff
f
mf
mp
p
fff
N
pp
ff
f
poco cresc ...
mf
f
ff
f
poco dim.

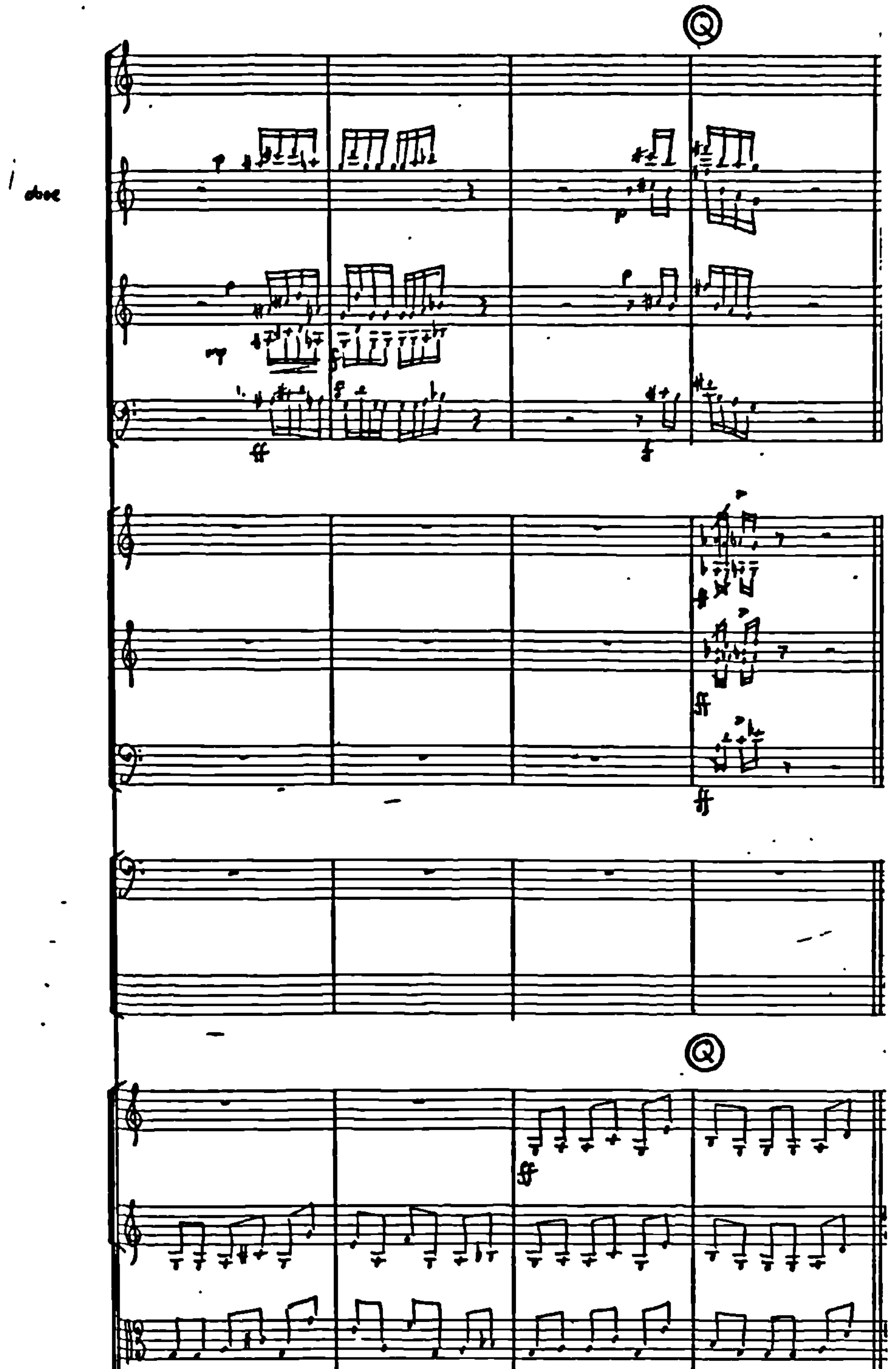


mf
p
pp
a.
2
poco
dim...
f
p
mp
ff


Ⓟ
piccolo 2.
mp
f
mp
2.
2.
Ⓟ
f
ff
pp
f
p
pp
pp
poco cresc . . . . .



PANOPUS SCORE-SYSTEM ©



PANOPUS SCORE-SYSTEM ®

Appendix B

TEXT YEARS (Score)

text years

stochastic texts for choir

the piece should be performed in a resonant
acoustic

the choir is split into four groups of readers

instructions for each reader:

1. beginning anywhere in each block, read the text at a normal pace, observing the gaps

2. where the text is spread down a whole column, you may read from any part of it; otherwise you read only from the part of the page with text for your particular group

3. on reaching the bottom of the block, go back to the top, and if necessary continue reading what you've already read in order to fill up the time

4. let pulse patterns emerge naturally
   don't aim to set up a pulse, but don't aim to destroy pulse either

5. a solid line at the end of a block: | indicates an instantaneous change
   a broken line between blocks: ⋮ indicates a gradual change from one to the other:

   finish the line you're on before going on to the next block (two or three seconds merging time)

Kevin Jones   Dec 1977

Text Years

Kevin Jones

1 2 3 4

10"

dum dum dum dumadum
dum dum dum dum dum
1 dum dum dumadum duma
dum dum dum dum
dumaduma dumadum dum
duma dum dum duma dum
dum dumadum dum duma
dum dumadumaduma dum
dum dum duma dumadum
dum dum dum duma dum
dum dumadum dum dum
2 duma dum dum dum
dum dumadum dum dum
dum duma dum dum
dumaduma dum dum
dumaduma dum duma dum
dumadum dum dum′ dum
dum duma dum
dumaduma duma dum
3 dumadum dum dumadum
dumadum dum duma
dumadum duma dum dum
duma dumadum
dumaduma dum duma
dum duma dumadum dum
dum dum dum duma dum
dum duma dum duma
4 dumadum dum duma dum
dumadumadumadumadum
dum dum duma dumadum
dum dum dum dum dum
dum duma dumaduma

10"

dum dum da dum dum dum
dum dum dumdumdum dum
dum da da dum duma
dumdum da dum dumdum
duma dumdum dumdum dum
dum duma dum da da da
dum duma dumdum da da
da dum dumdum dum
duma dumda dumdum dum
da dumdumdum
dumdumduma dum dum
duma dum dum dum dum
dumda da dum dum dum
duma da dum dum da
dum dum dum da dum
dumda dumdum dum da
dum dum dum da dum
dumdum dum da duma dum
dum dum da dumdum duma
dum da dum dum duma
dumdum dum dum da dum
da da da dum da dumdum
duma dumdum dumduma
dumdum dum dum dum dum
da dum dum dumdumdum
dumda da
dumdumdumdum dum da
duma dumdumdum da dum
dum dum dumdumdum
dumdum dum dum dum da
dumdumduma dum dum
dum dumdumdum dum

10"

dum da dum dadum dum
dam dum dadum dadum
dadum dam da dum dum
da dum da da dam
dadum dadam dum dum da
dum dadum da da
dumdadadadum dam da
dum dum dadadadum dum
dadum dadum da dum
dada dam dum da dum
dum dadadum dum dum
dum dumdum da dum dum
dum dum da dada dum
dum dumdumdadam da
dadam dum dum dumdum
dum dada dum da da
damda dum dum da
dadumdum dumdum da da
da dum dum dumdadum
dadadum dum dadum
dadum dum dum da dum
dum dadum dadumdada
dum dadam dum dumdum
da dum dadadum dadum
dum dum dum da da
dumda dada dada da da
dam dum da da dum
dadam dadum da dada
dadumdum dum dum dum
dam dam dum dadum dum
dumdadum dadumdum
dadumdum dadum dadum

10"

da da dada da da da
da dum dum da dada
dum dum da da dum da
da dum..da..dum dum dum
dada da dum da dum
dada da da da dada
dadadum da dum dum da
dadadada da dadada
dum dum da dum da dum
da da dum da da da
da da da dadadum dum
dum dadum dum
dadadum da dada da
dum da dada dadada
da dada da da da dum
dada *dum da dum da da*
da da da da dum da
da dum dum da da da
da da dum da dadadum
da da dada dada dada
da dum da da da dum
dadum dadadadadum
dada dadum dum da
da dum dadadada
dadum dum dum da
dadum dum da dada da
da da da da da da da
dum da dada dadada
dum da dadada dadum
dum dada da da da dum
dum dadum da dum
dadum da da dum da

ppp pp mp f ff

4 5 6 7 8

10" 8" 8" 10"

da da dada da da da
da dum da da da da dum
da dadada dum da da
dum da dum dum da da
dadada da dum da dum
 dada da da da dada
da da dada dada da da
dadadada da da da da
da da da dadada dum da
 da dum da da da da
da dada da dum dum
dum dada da da da da
da da dum dum da da
da da da da dada da da
da dum da dadum da da
 da da da da da da
dum da da dum dum da
da da da da dum da
dadadum da da dada
dada da da dum da
da da dadum dada
dada dadum dada da
da dum da da dum
dadada dum dum da da
da da da da da da da
da da dum da da da
da da da da dada dada
da dum da dada da da
dum da da da dum da
dum dum da dum da
da da da dum dum
dum da da dum dum

ff —— fff p

shotupititer a ster
shotupist ster ster a
stiter stit ster pit a
stit a pistist stister
ster a a ster a
shotiter ster a shoter
stupitistutupitupit a
a pitut a a stit stit

mf

shuba sha sha shub
shub sha shab shub
sha shubab shub shub
shub shub shuba shubaba
shabab shub shab sha
shaba shab shub
shubabab sha shub shab

shubababab shubab shub
shuba shub shabab shub
 shub shab shub shub
shuba sha shuba shub
 shub shub sha shabab
shab shub shub shab
sha shuba shab shuba

shub shubab shaba shub
shuba shuba shuba shub
sha shab shub sha sha
shub shuba shab shubaba
sha shuba shab shuba
shabab shuba shubab
shuba shaba shub shubab

8 9 10 11 12

10"

spitak spak spik ak
stita spak spak
stita spak cra spita
aspa craspa cra
aspak a akak spit
spak a spak stak ak
spak ak ak crak spa
stak spikakakak

f

shub shub shub
shubaba shubab shuba
sha shubab shub shub
sha shub shuba shuba
shub shub shubab shab
shub shab sha shub
sha shuba shub sha
 shuba shub shub

shub   shub shab
shub shubab shubaba
shubab shuba sha
shub shab shubabab
sha shuba shub sha
sha sha shub shub
shub shaba shubaba

p

10"

crakak crak claclak
kak crakakak kaka
 crak klakacrak
crakak cra cra
crakak cla craklakla
crakacra crakaklak
clacrak clacracclac-
rak claklaclaclacrak

f

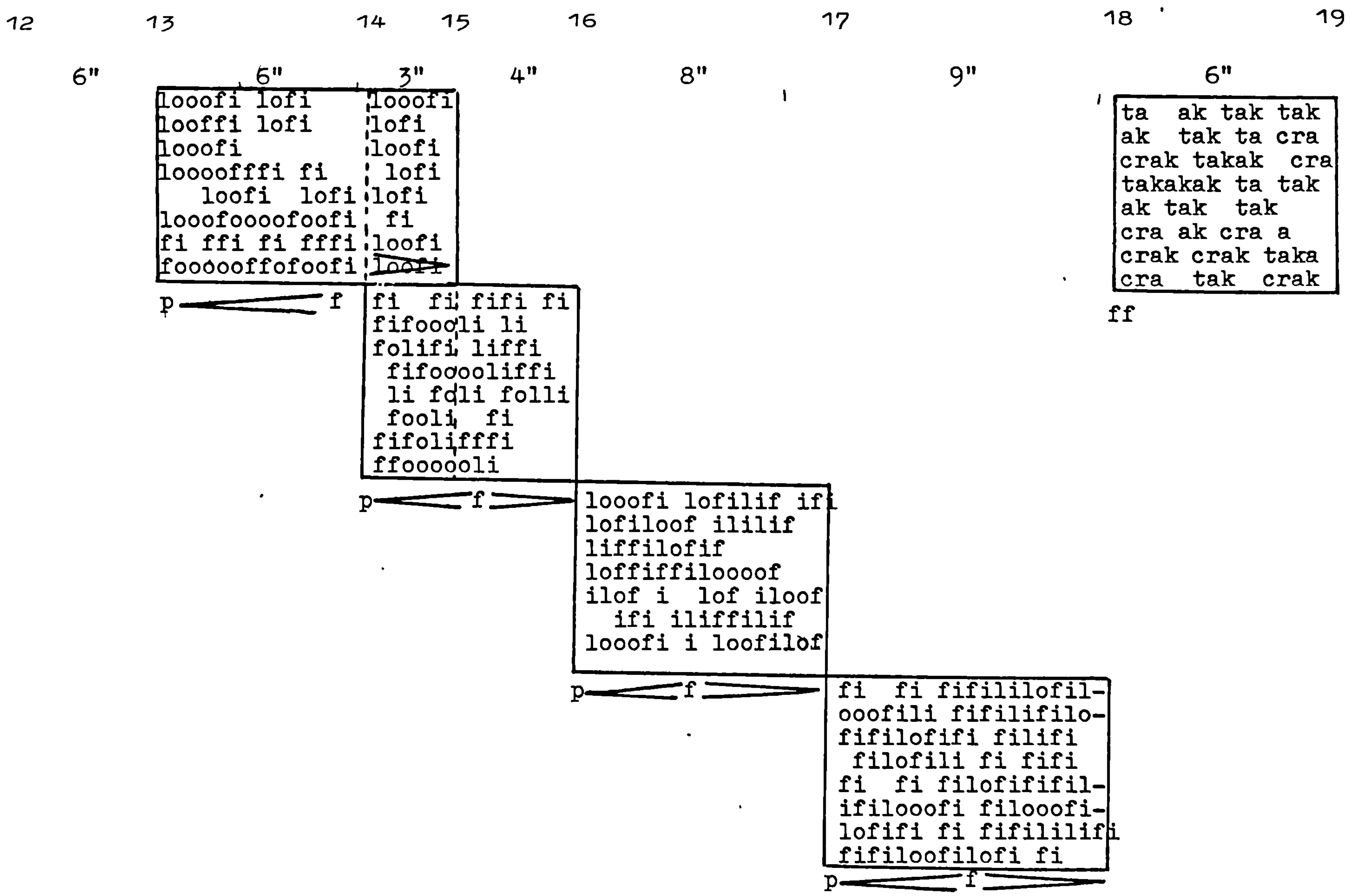
6"

22"

lora lubla tikikikitikitiubla looora la lubla
tikitikitititiubloora ubla lora lubla luba
looora lublubla uba lubla lora la loooora
lora lubla tikikikikikititikitikititikikiki-
ubla la lublora ubla looora lubla lublora
loora uba ta tiublubla lublora lubla la
lubla tikiublubla la la lubloooora
tikiublora ubla luba lubla lora la la lora
la tiubla luba   lora lubluba lublublubluba
lubla la la lora la la loora lubla la lora
 lubla ubluba la la lublooora ubla ublubla
tiubla la looora lublublora titikikitiubla
lubla uba lubla la la tiublublubla la la
loooora la la luba tikitikikikiubloooora
lora lubla titikiublubla ubla tikikikikiub-
looora tikikitiublora la ubla tiubla la
tikikikiublubluba ubluba loora lora
tikikiubla lubla lubla lubla la la la la
tiubla titiublora loora la la lubloora la
tikikitikikikikitikiubluba ublubla la
ublubla lora la titikiubluba lora lora lubla
loora lublora looora tikikikitikiuba la
tikitiuba tiubloora lora ubla tikikiubla la
loooooora la tiuba tikitiubloora uba la
lubla uba la lublublublubla tiubla la
tikititikitiubla lubloooora tititikitiuba
la looora la lubla ublubla lora lublublubla
luba looora la tiubla lora lubloora luba
lubla ublublora ubla lora la lubla lora uba
lubla la ubla tiubla la la   ublora uba
tikitikiubla loora lublublubla la lora
ublublublublubla la lublora loora lublooora

mf

3.



looofi lofi
looffi lofi
looofi
loooofffi fi
loofi lofi
looofoooofoofi
fi ffi fi fffi
fooooffofoofi

looofi
lofi
loofi
lofi
lofi
fi
loofi
~~loofi~~

fi fi fifi fi
fifooli li
folifi liffi
fifoooliffi
li foli folli
fooli fi
fifolifffi
ffoooooli

looofi lofilif ifi
lofiloof ililif
liffilofif
loffiffiloooof
ilof i lof iloof
ifi iliffilif
looofi i loofilof

fi fi fifililofil-
ooofili fifilifilo-
fifilofifi filifi
filofili fi fifi
fi fi filofififil-
ifilooofi filooofi-
lofifi fi fifililifi
fifiloofilofi fi

ta ak tak tak
ak tak ta cra
crak takak cra
takakak ta tak
ak tak tak
cra ak cra a
crak crak taka
cra tak crak

4

4

```
19      20      21          22     23          24     25       26        27
    5"      3"       6"         3"        6"       3"      4"       4"
```

```
takak  tak cra crak
tak cra    ta  tak
ta akaka crak a tak
ta a cra cra  crakak
a a crakaka akakak
takaka taka cra  taka
takakakak crak crak
takak tak tak tak ta
```
ff

ff
```
ta cra ta  tak crakak taka a ta
taka   tak crakak ta takaka tak
 cra  craka a crak tak ak ta takak
  ak  cra ta tak  a cra  a ta crak
 ak taka a  ta akak takakak aka
  takak crak ta  a crak crak ta
crakak ta ak akaka cra tak tak
tak  a ta cra   ta crak crak crak
```

```
crakak a  cra ta
  crak crak crak
 tak cra ta crak
tak  tak takak
ta ta   tak
crak ta  takaka
  cra crak taka
ta crak crak tak
```
ff

```
ak takakakaka
crak  ta crak
ta akakaka ta
tak cra  tak
taka takakaka
crak   tak akak
tak tak crak ta
craka  ta ak ta
```
ff

```
tak  tak  cra  tak
taka craka craka
  tak  taka crak
ta ak  crak tak
 crakaka  crak ta
 cra ta crakak
taka crak  crak
tak cra   ta taka
```
ff

```
 cra cra  tak ta  tak
   cra  cra  craka cra
 tak ta ta crak tak
taka crak crak tak cra
 ta cra  crak tak
crak tak a tak crak
taka  crak ak crak
  tak  taka crak cra
```
ff

27 28 29 30 31 32

6"

tit titipiti itit
pipititititi
titititi i ti pi
it pit itipipiti
ipi pipit pipit
tipipipipititi
titi ipit ipipip
titipitipi tit

mf

7"

tipi itipit ipit
ititit
titipipititititi
ti ipi tititi
titi ipipipipiti
tit   tipipitititit
ipi pipi pi it ti
titit it pit ti
tipi ipititi it
ipitipi tit
ipipitititipipi
tipi pipitit
pipititit tipit
ipipipi it tipiti
ipi ti pit   ti
pipitit iti ti

8"

pit it pipipitit
tipi itit pi
pitipiti pititi
pitipitipipit ipit
tipipititi
pititipit it it it
titi titi   ipipi
pit tit tipipi pit
titipipitit pititi
pipipi tipi
ipitit it ititit
pi pi pitit iti
pipititipipiti
itipi ti it titi
tipipitipi
tititipipi
tipitititit ipipi
it iti tit pit
pipiti pit tipi
ipit titi tipi ipi
tit tit pitipipi
tipit ipipipitipit
tit pitititit
pipiti pitipipititipi

9"

ipipit tipitipiti
tipipipiti   pipipit
pitit pit tipitit
tititipit itit
titipipitipi
ipipipipitititititi
iti ti tipit pipi
pipitipipipi ti
tipipit tit tit piti
pi pit pipit itititi
titipi tipipi itit
pipitipipipi tit
itipi pi tit ititipi
ipi itipiti ti tit
tit pititit tit titi
ti itipipit ititit
pi pipi ipiti ti
tipipit tit ipititi
pitipipitipipipi
tipit ititipit   ti
ipipipitipipitititi
piti pi it ti piti
it tipitipitipipipi
titpit tipititit
iti tipitipi tit
pipitipipititi ti
titipit ititipipi
tit titipit ipi piti
pitipipititipitipip
tipi tipi tipiti
ititi tit titipi
titi titipitipi pi

> pp

8"

di tip dipitip dip
tipiti dip
dipitipipitipip
tipitipi titip diti
di tip di titipip
dipipitip titipi
dip tipip dipi di
tipitipipidipiti
dip di ti pipipip
ti tip di ti ti
tipi dipipipi diti
ti dipipi
tititipip ditipipi
pip tip di ti ti
dip ti tipip dipi
tipi dip tipi
tipidipipitip
tipiti tipi diti
dipip dipipi dip di
ti titip ditipip
ti tipi dipi   di
dip di di dipi di
ti dipi tipipipipi-
pi tipitipipititi
dipipipi titip di
tip ti di dipip di
tipipipi tip di
dipi titi dipitip
dip ti tip diti
dip ti tipiti dip
tipi di di ditip
di di dip di tipi

32 33 34 35 36 37

8"	8"	8"	8"	8"
dip a dip dip a	dim dipa dim diba	dim di didimadidim	dum dima dimaduma-	dum dum dim duma-
dip a di dip dip	diba diba diba	didima didimadim	dum dim dum didim	dum dim dum dum
dip a dip a dip dip	diba dipa dim diba	didim didim dimadim	didum dimadim dima	dum dum dum dum
dip dipip dip tipi	dipa dim dibdipa	dima di didi di	duma dum dim dim	dumadum dumadim
di tip a dip a a	dipa dipa diba	dima dididadimadim	dumadima didimadum	dum dumadum
dip dip a ti dip a	diba diba	didi dim didim dida	duma dum dima	dumadumadimaduma-
dip a a dipipip	dibdibdipa diba	di dida dimadididim	duma dim	dum dum dum dum
a a a dipip a dip	diba dipa diba	da didima dadidada	didididididum dim	dum dimadum
a dipip dip a dip	dibdim dim	dimadimadim dim	dima dim dumadidi-	dimadumadumadum
dip dip dip tipip	dibdibdibdiba	dima dididima dima	duma dim dim dim	dum dum dumadima-
dip di di a a a	dibdibdiba dim	di dididim di	dima dumadima dim	dum dum dum dum
dip dip a a dip	dibdiba dipa diba	dimadidimadimadim	dum didim dim	dum dim dimadum
dip dip a a di	dim dipa diba	da da di dima dim	dima dim dum dim	dum dumadimadum
dipip ti dip dip	dibdiba diba dim	didim didi dima	dimadum dima	dum dum dum dum
a dip tip tip di	diba dipa dipa	dim dima	dum dum	dum dum dim
tipip a dip dip a	dim dipa diba	didimadima dimada	dumadimadidum dum	dumadum dum
dip dip tip	diba diba diba	dim da didim	dim didumadim	dumadum dum dum
tipipipi dip tip a	dibdim dipa	didididimadi dim	dum didima dim	dum dumadumadum
dip dipip a dip	dibdiba dibdim	dim dima di dim	dum dumadum	dum dim dumaduma-
dipi dip a a ti a	diba diba dim	dim da dima didi	didima duma dim	dum dumadum dim
dipi a dip dip a	diba dipa dipa	dima didimadim	didim dim dim dim	dum dum dumadim
dip a di dip dip a	diba dipa dibdiba	di dima dada dima	dim duma duma	dum dum dim dum
a tipip a dip di a	dibdibdiba diba	dima di dida dima	duma dima dima	dumadumadum dum
dip dip dipi a dip	dim dipa dibdim	di dimadima dim	dimadidumadimadi-	dim dumadim duma-
dipip a a a a dip	dibdiba diba diba	didi dadima da	didum didimadim	dum dumadum dum
dip tip ti di dip	diba dipa dibdiba	dadadidi didma dim	dim dum dima dum	dumadum dum
tip tip a di a a	dibdibdim dibdim	di dima dim di	dum didum	dumadumadum dum
dip dip a dip a	dipa dibdibdibdiba	dada di dim di dim	didumadidum didim	dum dim
dip dip dip tip di	dibdim diba dim	dim dida di didi	dim duma dimadum	dumadumadim dum
di dip a dip dip	dipa dipa dipa	dima dim di da da	dimadiduma didum	dim dum dum
dip dip dip dipip	diba diba	di dima dima dimada	dim dum diduma	dumadum
dip a a dipip a dip	dibdibdiba dim	dida dima dima	didim didumadim	dumadum dumadum

pp p mp mf f ff

7

```
37                       38                39                  40                41                42

          12"                      8"               8"                 7"               6"

dum dum  dum  duma dum                    hiss hissishis his ishisshis ishis ishis isshis
dum dumadum duma dumadum                   sssishiss hissshisis  hisiss hishisssshis
dumadum duma duma                         hishis sshis ishishiss hishishishisss shis hisss
dumaduma dum dumadum  dum                 ~~hishishiss sis is~~ ssshis hisiss hiss  hisshis is
dum duma dum  dum dumadum                 sss~~Ishis his~~ his shishis his   his hishishis
dumadum  duma duma dum dum                his  sis isissss shissis his his  hisis isishis
dumadumadum duma duma                     isshiss  his iss shissis hishis hiss hiss hiss
dumadum dum dumadum dum                   sishisss hishissshissshis  his isissss  his isss
dum duma duma dum dum duma                his  shisss hishishishis shis is  hisisis
dum  dum   dumadum dum                     shishis is his  is  his sshis is sshis isis his
dum dumadum dum   dum  duma               is is iss is is iss hisshissishisiss hiss shisis
 dum dumadum dum                          isiss  issisis  hishis his ssssis shis
dumadumadum dum dum dum                   hisishishisss  siss hisis is  sisis  shiss
dum dum dum duma  dum                     isshiss hisishis  sshis  hisshis hishishiss  his
dumadumadumadum dum dum                   his is ishiss his is hiss hishis ssis his
duma duma dumadumadum dum                 ssshisshishishis shissis ishisshississss
  dum dum  dum dum dum dum                hisshishishis sis is isis isiss  ssis  hiss shis
dumadum dumadumadumadum                   shiss hisss  is isissis shis  his hishisshis
dum dum  dum duma                         shiss isshissss is ishisisis  isshis isishisisss
dumadumadumadum dum  dum                     shis shiss his shis isss is  isisiss  is  is
dum dumadum dumadum dum                   isishis shis his hisss sssssshis hi~~sisisshisss~~
dum dum dum dum                           his his sis  ishiss isis shis is shis  ~~hishis~~
dumadumadumadum dum dum                   isss isis    is  is hiss hishisss shishis
dum dum dum  dum dum dum                  ssissis hiss issisis his  shisshis is  sis
dumadum dum duma duma dum                 hissshis his  ishishisshiss shisss is isss
dumadum duma dum dumadum                  sssiss hisishis  hissis is ishishis isss ssishis
dum dum dum dum  dum dum                  sis hisss hishisishisshis sis hisssisss his
dum dum duma   dum dum                    ississsshishishis ishishis is his shis isishis
dum dum dum dum dum dum                   issshis  sshishisisis ~~sshiss ssis~~his his
dum dum  dumaduma dumadum                 issshis  sis hiss~~hissshisiss~~ shis  hissshis
dum dumadum dumaduma dum                  hishishis hisshis  is hishis isiss is  ishishiss
dumadum dum dumadumadum                   ssis isshishishishiss hissis hisss his is is

ff                      fff               p
```

8

18

```
42                    43                        44         45         46          47

         9"                       10"                5"         8"                    6"
shisis  hishishishis  ishiss iss is  isis
shis  is sshisshisishi sshis is  shis   hishis-
shishis  his shissssh isisis his his ishis
ssshishis isishissshis shisis  issss his
hisshis hisss hisss is hisishishishishisiss
sis sssshis is hiss·ss issss shis sssis shiss
shis iss his isis his   is  shis ishiss   is
shishishisshis sis is     his  his sshishiss
ishis shiss iss shis s ishissis sis is ishiss
his isiss  ishiss iss  shishishishisss  is
hissis shis ishiss isi s hiss  is is
ishisissssisishis shis sis isis sshis sshis is
his shisishishishis hi s iss sssis shis shis
hiss shishishis iss hisis hishis his   is  his
  ishisshis ssshis hi s  is his shis hisisiss
ishiss shis ishis hish isis  his hishishisss
is is  is shiss ssis s is shisishis ssshis
sshisss  hisis  sis si s his is shisiss
hisissshishis   sshi s hishiss ishisishis
hisss hiss is issshis shiss hishis ishissshis
shisshiss  isishishis  is isshiss ishisisissh-
isis hisss ishisis hi sshishis  ishis his
hishiss  ssisis sis  shishis  ishis   hiss
hiss sis isissshishis s  shiss shishis hiss
his his  his iss his hisishishissishisshisssh-
isss his shishiss shi s shiss   his issisisss
 issshiss hisiss isis hishisssisiss is is hiss
  sis shishiss  is  hi sshishis hishis  ishis
hisisisss  hisishisis s is shishisis sishis
sshis  hisss siss sss ssishiss isiss  ssis
hissshisshis hishishi s sishishishishishis-
shishis is is  hishi shis ssshis   sishis
                                         ppp

ra ra  rara ra heri
 ra ra rariheera
heera rari ra rara
ri ra hera he rari
he ra ra
riheeeeerariheriha
hee rara ra ha he
rara heeee heera ra
mf

fliga  flig flig
fliga flig a
flig flig fliga
flig atig flig
a flig tiga  a
tig tig  a atig
flig flig a
fliga fligatig
mp
```

47 48 49 50 51

12"

dop adoopoopoopadop spoop asp
adoop adooo asp spoopooop
aspo a adopaspooopaspadooo-
opo adoo spooopopadodo
adodoodo aspoop asp
adopopadooodop aspoopoop
spopopodoopadoooopoop
spopaspop asp adoopadopasp

pp

7"

fit fi fifitit fit
ititit fifititifit
fifit fifit fititifit
itififiti fi fifi
fi ifititi
fififitifit itififit
ititi fifi fi
fifititifi fifit

pp

6"

10"

along along abong abong
abongabong abong
alongabong bong alonga
bong abong bong along
abong alonga bong abong
alongabong along bong
abong abongabong abong
bong abong bong abong
bong abong abong abong
abongabong abong abong
along abong abong bong
bong abong alongabong
bong alonga abong abong
bong bong abong abong bong
abongabongalonga abong
alonga bongabong abong
abong along abong along
bong abong bongabong
alongabong bong bong bong
abong abong bong along
abong abong abongabong
abong bong bong bong bong
abong abongabongabong
along abong abong along
bong bong bong bonga
abong abong bong along
abong abong along along
abong bong a-ong abong
abong abong along along
bong along abong abong
bong abonga abong abong
abongabong alongalongabong

pp

10 10

51 52 53 54 55

10"	10"	10"	10"
along along along bong	bonga bong dong donga	konng kong klong long	konng kong klong
bongong ag bong along	dong bong dong donga	kong kongong long ag	kong konng kong
bong abong agong bong	dong bong dong dong	long klonnng long	klonnnng kong gonnnnng
bongong along	klong klong donga dong	konngagonga konngong	kong long lonng
alongongong bong along	dong bonga donga	long longa kong gag	konngong klonnnng
ag gongalong bongagong	dong dong bong donga	kong ag klong agag	kongong kong kong
along gong ag ag	donga dong dong dong	longong long kong	gonng kong kong
abongabong agongong	klong bong dong dong	kongag kong konng long	kongongong klong
bong bong gagongongong	donga dong dong klonga	klong ag long lonnga	klonnnnng kong kongong
alongongong ag along	dong dong dong dong	long long konng	lonnnnnnnng konnnng
along along abong ag	dong bong bong donga	agonngong kong klonng	klong klonng konng
along bongongong	bong klong dong dong	agong konngag klong	konng konnnnnng kong
abongalong bong	bong bonga klonga dong	kong longonng gag long	klonng konnng long
abongong alongabong	dong klong bonga	lonngong long long	gonng konnngonng
bong alongongongalong	klong dong klonga dong	ag long lonng klonng	klonng kong klonng
gong ag gongongongag	bong klong dong bong	longa long klonga long	konnnng klong konng
gong bongong along	donga dong dong klong	klonng klong long	klong konng long kong
alongong gabong along	bong dong klong klonga	klong longagaga long	lonng klonnnng gonng
alongalongong abong	donga dong dong donga	lonng kong long	kong longonng konnng
bong agong bong	dong bong dong bonga	lonngongaga konng	konnnnng klong
alongong bongong abong	bonga dong bong bong	long longa klong long	lonngongong long
abongong alongag bong	bong donga klong dong	long ga ga long	konnnng long konnnng
alongong	klong dong dong dong	longong long klonng	gong konng konnnnng
bongagalongong ag gong	klong dong klong bong	klongonngag ag lonng	klonngonnng long
alongong bong abongong	bongaklong bong dong	kong klong long konga	kongongonnnng konng
gong ag abong bong	bong dong bonga bong	longonng long longonga	kong klonngonng kong
bongong ag abongong ag	dong donga dong bong	konnngong kong gong ag	long gonnng konnng
alongongabong bongong	dong donga dong bong	longonnnga long gong	klong konnng kongonng
ag ag ag gongong	dong dong klong	longong long longa	klonnnng long lonng
agalong along	klong donga dong dong	lonngonga long klong	konnnng konngong long
agongongongalong along	bonga donga dong dong	agagong klonnnng	kong long klonngong
ag alongong ag ag	bong donga donga dong	kong klongagongag	klong long long kong

fff

END

Appendix C

Text Analysis and Synthesis Program

```
'BEGIN'
   'COMMENT' PROGRAM TO SCAN TEXT AND PRODUCE
             A SYNTHETIC TEXT BASED ON TRANSITION
             PROBABILITIES. DATA CONSISTS OF
             TEXT INPUT (END OF LINES AND MULTIPLE
             SPACES ARE IGNORED) WHICH SHOULD
             TERMINATE WITH / , FOLLOWED BY THE
             NUMBER OF SYNTHETIC WORDS REQUIRED;

   'INTEGER'I,J,JC,WORDS,TOTALWORDS,JC2;
   'REAL'X,SUM,R;
   'REAL''ARRAY'ST[1:27,0:27];
   'REAL''PROCEDURE'RNDEC(B);
      'COMMENT' PRODUCES A RANDOM REAL NUMBER, STORED
         AS X, BETWEEN 0 AND 1. X MUST BE SET TO AN
         ARBITRARY VALUE TO BEGIN WITH;
      'VALUE'B;'REAL'B;
      'BEGIN'
         'COMMENT' GIVES X A NEW VALUE;
         X:=1.9173*X+B;
         'COMMENT' MAKE X A DECIMAL VALUE BETWEEN 0 AND 1
            BY REMOVING THE INTEGRAL PART OF THE NUMBER;
         RNDEC:=X:=100*X-ENTIER(100*X);
      'END';

   'COMMENT' INITIALIZATION;

   'FOR'I:=1'STEP'1'UNTIL'27'DO'
   'FOR'J:=0'STEP'1'UNTIL'27'DO'
   ST[I,J]:=0;
   WORDS:=0;

   'COMMENT' READ DATA AND CONSTRUCT MATRIX;

   'COMMENT' I IS SET TO THE VALUE FOR A SPACE TO BEGIN;
   I:=27;
NEXT:
   JC:=READCH;
   'COMMENT' DEAL WITH SPACE AND END OF LINE CHARACTERS;
   'IF'JC=CODE('('/')')'THEN''GOTO'DO;
   'IF'JC=CODE('('EL')')'THEN''GOTO'NEXT;
   'IF'JC=CODE('('%')')'THEN'
   'BEGIN'
      JC2:=NEXTCH;
      'IF'JC2=JC'THEN''GOTO'NEXT;
      J:=27;
   'END'
   'ELSE'
   'COMMENT' ADJUST THE RANGE OF INTERNAL CHARACTER VALUES
      TO THE MATRIX DIMENSIONS;
   J:=JC-3872;
   ST[I,J]:=ST[I,J]+1;
   I:=J;
   'GOTO'NEXT;
DO:'FOR'I:=1'STEP'1'UNTIL'27'DO'
      'COMMENT' CONSTRUCTS CUMULATIVE PROBABILITIES;
   'BEGIN'
      SUM:=0;
      'FOR'J:=1'STEP'1'UNTIL'27'DO'
      SUM:=SUM+ST[I,J];
      'IF'SUM'NE'0'THEN'
      'FOR'J:=1'STEP'1'UNTIL'27'DO'
      ST[I,J]:=ST[I,J]/SUM+ST[I,J-1];
   'END';

   'COMMENT' PERFORM;

   X:=0.31977;
   TOTALWORDS:=READ;
   I:=27;
LETT:
   R:=RNDEC(4.571);
   'FOR'J:=1'STEP'1'UNTIL'27'DO'
   'IF'R<ST[I,J]'THEN''GOTO'SKIP;
SKIP:
   'IF'J=27'THEN'
   'BEGIN'
      WRITETEXT('(' ')');
      WORDS:=WORDS+1;
      'IF'WORDS/6=WORDS'/'6'THEN'NEWLINE(1);
   'END'
   'ELSE' PRINTCH(J+3872);
   I:=J;
   'IF'WORDS=TOTALWORDS'THEN''GOTO'FIN
   'ELSE''GOTO'LETT;
FIN:
'END';
```

APPENDIX D. COMPOSING PROGRAMS USED IN "MACRICISUM"

Appendix D1. GRAND

```
BEGIN
   INTEGER I;
   REAL S,T,R;
   LABEL SKIP;
   BOOLEAN FLAG;
   OPEN(1,"DSK",0,0,2,0,0,0);
   ENTER(1,"RAND",FLAG);

   CPRINT(1,
   "INSTRUMENT TOOT;
    OSCIL(P4,MAG/P2,P5);
    OSCIL(U1,MAG*P3,P6);
    OUTA_OUTA+U2*P7;
    OUTB_OUTB+U2*(1-P7);
    END;
    NCHNS_2;
    ARRAY F1,F2,F3,F4(512);
    SEG(F1);0,1  1,20  .5,60  0,100;
    SEG(F2);0,1  1,15  ,5,30  .5,75  0,100;
    SYNTH(F3); 1,1  2,.9  3,.8  4,.4  6,.3  8,.2  999;
    SYNTH(F4); 1,1  3,.7  5,.5  7,.5  9,.5  11,.3  999;
    PLAY;
   ");
   FOR I_1 STEP 1 UNTIL 40 DO S_RAN(0);
   S_0;
   FOR I_1 STEP 1 UNTIL 60 DO
   BEGIN "LOOP"
      T_RAN(0)*2;R_RAN(0);
      IF T<0.1  THEN T_0.1 ;IF R<0.1  THEN R_0;
      CPRINT(1,"TOOT" ,S,T,RAN(0)*RAN (0)*7000,RAN(0)*100+300,
      IF RAN(0)>0.5 THEN "F1" ELSE "F2"," ",
      IF RAN(0)>0.5 THEN "F3" ELSE "F4",
      R,";
   ");
      S_S+T+RAN (0);
      IF S>20 THEN GOTO SKIP;
   END "LOOP";
SKIP:CPRINT(1,"FINISH;");
   CLOSE(1);
   RELEASE(1);
END
```

Appendix D 2. GSTOK

```
BEGIN
   INTEGER I,J;
   INTEGER ARRAY V[1:9];
   REAL ARRAY P[1:9,1:2];
   BOOLEAN FLAG;
   OPEN(1,"DSK",0,0,2,0,0,0);
   ENTER(1,"STOK",FLAG);

   CPRINT(1,
   "INSTRUMENT TOOT;
    OSCIL(P4,MAG/P2,P5);
    OSCIL(U1,MAG*P3,P6);
    OUTA_OUTA+U2*P7;
    OUTB_OUTB+U2*(1-P7);
    END;
    NCHNS_2;
    ARRAY F1,F2,F3,F4(512);
    SEG(F1);0,1  1,20  .5,60  0,100;
    SEG(F2);0,1  1,15  .5,30  .5,75  0,100;
    SYNTH(F3);1,1  2,.9  3,.8  4,.4  6,.3  8,.2  999;
    SYNTH(F4);1,1  3,.7  5,.5  7,.5  9,.5  11,.3  999;
    PLAY;
   ");
   FOR I_1 STEP 1 UNTIL 7 DO J_RAN(0);
   FOR I_1 STEP 1 UNTIL 9 DO
   FOR J_1 STEP 1 UNTIL 2 DO
   P[I,J]_RAN(0);
   FOR I_1 STEP 1 UNTIL 3  DO J_RAN(0);
   FOR I_1 STEP 1 UNTIL 9 DO
   V[I]_RAN(0)+1;

   FOR I_0.1 STEP 1 UNTIL 7  DO
   BEGIN "I"
      FOR J_1 STEP 1 UNTIL 9 DO
      IF RAN(0)<P[J,V[J]] THEN V[J]_1 ELSE V[J]_2;
      FOR J_1 STEP 1 UNTIL (V[3]+1)*15 DO
      BEGIN "J"
                          J_J+RAN(0)*V[4];
        CPRINT(1,"TOOT ",I+J/15,(IF RAN(0)<0.5 THEN "0.06" ELSE "0.05"),
               (375*V[2]-350)*(3-V[1])*(2^(V[1]+1)*RAN(0)+1),
               1000*V[5]-500,
               (IF V[7]=1 THEN " F1 " ELSE " F2 "),
               (IF V[8]=1 THEN " F3 " ELSE " F4 "),
               V[9]-1,";
               ");
      END "J";
      IF V[6]=2 THEN I_I+1;
      I_I+V[3]+1;
   END "I";
   CPRINT(1,"FINISH;");
   CLOSE(1);
   RELEASE(1);
END
```

Appendix D3. GRANP

```
3IN
 INTEGER I,J,K,LAYER,LENGTH,PITCH;
 INTEGER ARRAY PARTIAL[1:20],V[1:6];
 REAL ARRAY AMP[1:20],P[1:6,1:2];
 BOOLEAN FLAG;
 OPEN(1,'DSK',0,0,2,0,0,0);
 ENTER(1,'RANP',FLAG);

 LAYER_2;

 CPRINT(1,'INSTRUMENT STER;
  OSCIL(P4,MAG/P2,P5);
  OSCIL(U1,MAG*P3,P6);
  OUTA_OUTA+U2*P7;
  OUTB_OUTB+U2*(1-P7);
  END;
  NCHNS_2;
  ARRAY F2(512);
  SEG(F2); 1,1  1,100;
  PLAY;
  ');
 J_RAN(0);
 FOR I_1 STEP 1 UNTIL 6 DO
 FOR J_1 STEP 1 UNTIL 2 DO
 P[I,J]_RAN(0);
 FOR I_1 STEP 1 UNTIL 6 DO
 V[I]_RAN(0)+1;
 FOR I_1 STEP 1 UNTIL LAYER DO J_RAN(0);

 FOR I_0.1 STEP 1 UNTIL  90 DO
 BEGIN 'I'
    PITCH_RAN(0)*RAN(0)*400+30;
    PARTIAL[1]_15*RAN(0);
    AMP[1]_1;
    FOR K_2 STEP 1 UNTIL 20 DO
    BEGIN
       PARTIAL[K]_PARTIAL[K-1]+10*RAN(0);
       AMP[K]_RAN(0);
       IF AMP[K]<0.1 THEN AMP[K]_0.1;
    END;

    FOR J_1 STEP 1 UNTIL 6 DO
    IF RAN(0)<P[J,V[J]] THEN V[J]_1 ELSE V[J]_2;
    LENGTH_180*(RAN(0)+V[1]-1);
    FOR J_LENGTH/10 STEP 1 UNTIL LENGTH*0.9 DO
    BEGIN 'J'
       CPRINT(1,'SYNTH(F3);');
       FOR K_1 STEP 1 UNTIL 20 DO
       CPRINT(1,PARTIAL[K],',',AMP[K],' ');
       CPRINT(1,'999;
       ');

       CPRINT(1,'STER ',I+J/20,' 0.05 ',PITCH,
              1200/LENGTH*(IF V[2]=1 THEN J ELSE LENGTH-J),' F2 F3 ',
              (IF V[3]=1 THEN J/LENGTH ELSE 1-J/LENGTH),';
       ');
```

```
         FOR K_1 STEP 1 UNTIL 20 DO
         BEGIN 'K'
            PARTIAL[K]_PARTIAL[K]+
            (IF V[4]=1 THEN 2*RAN(0)-1
                       ELSE IF V[5]=1 THEN RAN(0)
                                      ELSE -RAN(0));
            IF PARTIAL[K]<1 THEN PARTIAL[K]_1;
            AMP[K]_AMP[K]+(IF RAN(0)>0.5 THEN 0.1 ELSE -0.1);
            IF AMP[K]<0.1 THEN AMP[K]_0.1;
            IF AMP[K]>1 THEN AMP[K]_1;
         END 'K';
      END 'J';
      I_I+LENGTH/20+7.5*(RAN(0)+V[6]-1);
   END 'I';
   CPRINT(1,'FINISH;');
   CLOSE(1);
   RELEASE(1);
END
```

Appendix D4. GRHY

```
BEGIN REAL SPEED,POS;
   INTEGER I,J,PITCH,    LAYER,LENGTH,RH1,RH2,RH3,        VOL;
   INTEGER ARRAY V[1:10];
   REAL ARRAY FP[1:10,1:2],RP[1:2,1:2];
   BOOLEAN FLAG;
   OPEN(1,'DSK',0,0,2,0,0,0);
   ENTER(1,'RHY',FLAG);

   LAYER_4;

   CPRINT(1,
   'INSTRUMENT NOIS;
    OSCIL(P4,MAG/P2,P5);
    RANDH(MAG*P3/16,MAG*P3*P4);
    OSCIL(U1,MAG*P3+U2,P6);
    OUTA_OUTA+U3*P7;
    OUTB_OUTB+U3*(1-P7);
    END; NCHNS_2;
    ARRAY F2(512);
    SEG(F2); 0,1  1,12  .5,50  0,100;
    PLAY;
   ');

   FOR I_1 STEP 1 UNTIL 10 DO
   FOR J_1 STEP 1 UNTIL 2 DO
   FP[I,J]_RAN(0);
   FOR I_1 STEP 1 UNTIL 10 DO
   V[I]_RAN(0)+1;
   FOR I_1,2 DO FOR J_1,2 DO
   RP[I,J]_RAN(0);
   FOR I_1 STEP 1 UNTIL LAYER  DO J_RAN(0);
   RH1_RAN(0)+1;RH2_RAN(0)+1;
```

```
FOR I_0.1 STEP 1 UNTIL 100 DO
BEGIN 'I'
   FOR J_1 STEP 1 UNTIL 10 DO
   V[J]_(IF RAN(0)<FP[J,V[J]] THEN 1 ELSE 2);
   SPEED_0.075*(RAN(0)+V[10]-1)+0.1;
   LENGTH_10*(RAN(0)+V[2]-1)/SPEED;
   PITCH_50*(2^(4+V[8])*RAN(0)*RAN(0)+V[8]^4);
   VOL_500+1000*(V[6]-1);
   POS_0.45*(RAN(0)+V[3]-1)+0.1;

   FOR J_1 STEP 1 UNTIL LENGTH DO
   BEGIN 'J'
      RH3_RH2;
      RH2_(IF RAN(0)<RP[RH1,RH2] THEN 1 ELSE 2);
      RH1_RH3;CPRINT(1,'NOIS ');
      CPRINT(1,I+J*SPEED,SPEED,PITCH,' ',
       (IF V[5]=1 THEN VOL
                  ELSE 1500/LENGTH*(IF V[6]=1 THEN J ELSE LENGTH-J)),
      ' F2 F3 ',
      IF V[2]=1 THEN POS
                ELSE (IF V[4]=1 THEN J
                                ELSE LENGTH-J)/LENGTH*0.9+0.1,';
   ');J_J+RH2;
   END 'J';
   I_I+V[9]+LENGTH*SPEED;
END 'I';
CPRINT(1,'FINISH;');
CLOSE(1);
RELEASE(1);
```

Appendix D5. GFM

```
GIN
 INTEGER I,J,K,PITCH,LENGTH,LAYER,TOTLEN;
 LABEL STOP;
 BOOLEAN FLAG;
 OPEN(1,"DSK",Ø,Ø,2,Ø,Ø,Ø);
 ENTER(1,"FM",FLAG);

 LAYER←1; TOTLEN ← 15;

 FOR I ← 1 STEP 1 UNTIL 5Ø DO
 BEGIN
    CPRINT(1,"
    INSTRUMENT FM",I,";
    OSCIL(P4,MAG/P2,P5);
    INTFP(Ø,1Ø*P2*MAG,P7);
    OSCIL(U2,P2*MAG,P5);
    NOSCIL(U1,U3+P3*MAG,P6);
    OUTA ← OUTA+U4*P8;
    OUTB ← OUTB+U4*(1-P8);
    END;");
 END

 CPRINT(1,"
    NCHNS ← 2;
    ARRAY F1,F2,F3(512);
    SYNTH(F3); 1 1 999;
    SEG(F1); Ø,1  1,5  .6,55  Ø,1ØØ;
    SEG(F2); Ø,1  1,1ØØ;
    PLAY;");

 FOR I ← 1 STEP 1 UNTIL LAYER DO J ← RAN(Ø);

 FOR K ← 1 STEP 1 UNTIL 4Ø DO
 FOR I ← 1 STEP 1 UNTIL TOTLEN DO
 BEGIN "I"
    PITCH ← RAN(Ø)*RAN(Ø)*6ØØØ+5Ø ;
    LENGTH ← (1Ø-PITCH↑Ø.5/8);
    FOR J ← 1 STEP 1 UNTIL RAN(Ø)*1Ø DO
    BEGIN "J"
       CPRINT("
       FM",K," ",I," ",LENGTH," ",
       PITCH," ",RAN(Ø)*1ØØ+1ØØ," F1 F3 F2 ",
       RAN(Ø)*Ø.9+Ø.1,";");
       I ← I+LENGTH+RAN(Ø)*LENGTH*(TOTLEN-I)/2;
       IF I > TOTLEN THEN GOTO STOP;
    END "J"; STOP;
 END "I":
 CPRINT(1,"
    FINISH;");
 CLOSE(1);
 RELEASE(1);
```

Appendix D5. GGLIS

```
IN
INTEGER I,J,K,PITCH,LENGTH,LAYER;
INTEGER ARRAY V[1:8];
REAL ARRAY P[1:8,1:2];
BOOLEAN FLAG;
OPEN(1,"DSK",Ø,Ø,2,Ø,Ø,Ø);
ENTER(1,"GLIS",FLAG);

LAYER_2;
FOR I_1 STEP 1 UNTIL 1 DO J_RAN(Ø);
FOR I_1 STEP 1 UNTIL 8Ø DO
BEGIN
   CPRINT(1,"INSTRUMENT GL",I,";
     OSCIL(P4,MAG/P2,P5);
     OSCIL(MAG*P7-MAG*P3,MAG/P2,P8);
     OSCIL(U1,MAG*P3+U2,P6);
     OUTA_OUTA+U3*P9;
     OUTB_OUTB+U3*(1-P9);
     END;");
END;
CPRINT(1,"NCHNS_2;
     ARRAY F3,F4,F5,F6,F7,F8(512);
     SYNTH(F3);1,1  3,.6  5,.4  7,.3  9,.25  11,.2  13,.1  999;
     SYNTH(F4);1,1  2,.6  4,.4  6,.3  8,.25  1Ø,.2  12,.1  999;
     SEG(F5); Ø,1  .8,7  1,12        Ø,1ØØ;
     SEG(F6); Ø,Ø  .3,5Ø  1,1ØØ;
     SEG(F7); 1,Ø  .3,5Ø  Ø,1ØØ;
     SEG(F8); Ø,1  1,95  Ø,1ØØ;
     PLAY;");
FOR I_1 STEP 1 UNTIL 8 DO
FOR J_1 STEP 1 UNTIL 2 DO
P[I,J]_RAN(Ø);
FOR I_1 STEP 1 UNTIL 8 DO
V[I]_RAN(Ø)+1;
FOR I_1 STEP 1 UNTIL LAYER DO J_RAN(Ø);

FOR K_1 STEP 1 UNTIL 8Ø DO
FOR I_1    STEP 1 UNTIL 2Ø DO
BEGIN
   FOR J_1 STEP 1 UNTIL 8 DO
   V[J]_(IF RAN(Ø)<P[J,V[J]] THEN 1 ELSE 2);
   CPRINT(1,"
     GL",K," ",I," ",LENGTH_ 5*(RAN(Ø)+V[1]-1)," ",
    PITCH_(21-1)*1Ø*(2^V[2]*RAN(Ø)*RAN(Ø)+V[2])+25,
     " 2ØØ ",IF V[3]=1 THEN " F5 " ELSE " F8 ",
     IF V[4]=1 THEN " F3 " ELSE " F4 ",
     PITCH+PITCH*1.5*(RAN(Ø)+V[5]-1),
     IF V[6]=1 THEN " F6 " ELSE " F7 ",
    .45*(RAN(Ø)+V[7]-1)+.1,";");
   I_I+LENGTH+(IF V[8]=1 THEN Ø ELSE 5);
END;
CPRINT(1,"
FINISH;");
CLOSE(1);
RELEASE(1);
ID
```

Appendix E : Programs and output for application of splitting grammars.

Appendix E1. The SPC1 program for generating shaded rectangles.

```
'BEGIN'
   'COMMENT' SPC1:PROGRAM TO DEMONSTRATE BINARY-SPLITTING
                  GRAMMARS.
                  SHADED VERSION WITH RANDOM ELEMENTS.

                  DATA IN THE FORM:

                  - LIST OF SYMBOLS ( .:-!+?%*$£),
                  - DIMENSIONS OF FRAME (120 X 60 MAX.)
                  - NUMBER OF FRAMES REQUIRED
                  - LIST OF DEFINING PROBABILITIES FOR EACH
                    FRAME. IN THE FORM:

                      PH   PV   PS

                    WHERE THE PROBABILITIES ARE:
                      PH - OF A HORIZONTAL SPLIT
                      PV - A VERTICAL SPLIT
                      PS - OF STAYING PUT

                    PH+PV+PS  MUST EQUAL 1;

   'REAL' X,PH,PV,PS,R;
   'INTEGER' WIDTH,DEPTH,I,J,K,FRAMES,L,CT;
   'INTEGER''ARRAY' RECT[1:120,1:60],SHADE[1:11];
   'REAL''PROCEDURE'RNDEC(B);
      'REAL'B;
      'BEGIN'
         X:=3.9177137*X+B;
         RNDEC:=X:=100*X-ENTIER(100*X);
      'END';
   'PROCEDURE' FILL(LH,UH,LV,UV);
      'INTEGER'LH,UH,LV,UV;
      'BEGIN'
         R:=RNDEC(3.5171341791);
         'IF' R>PV+PS 'THEN'
         'BEGIN'
            CT:=CT+1;
            'IF'CT>20'THEN''GOTO'SKIP;
            FILL(LH,(LH+UH)/2,LV,UV);
            FILL((LH+UH)/2+1,UH,LV,UV);
         'END'
         'ELSE''IF'R>PS'THEN'
         'BEGIN'
            CT:=CT+1;
            'IF'CT>20'THEN''GOTO'SKIP;
            FILL(LH,UH,LV,(LV+UV)/2);
            FILL(LH,UH,(LV+UV)/2+1,UV);
         'END'
         'ELSE'
SKIP:    'BEGIN'
            L:=ENTIER(RNDEC(9.3171833)*11+1);
               'FOR'I:=LH'STEP'1'UNTIL'UH'DO'
               'FOR'J:=LV'STEP'1'UNTIL'UV'DO'
               RECT[I,J]:=L;
         'END';
      'END';

      X:=0.717717391;
      CT:=0;
      'FOR'I:=1'STEP'1'UNTIL'11'DO'
      SHADE[I]:=READCH;
      WIDTH:=READ;DEPTH:=READ;FRAMES:=READ;
      'FOR'K:=1'STEP'1'UNTIL'FRAMES'DO'
      'BEGIN'
         PH:=READ;PV:=READ;PS:=READ;
         CT:=0;
         FILL(1,WIDTH,1,DEPTH);
         'FOR'J:=1'STEP'1'UNTIL'DEPTH'DO'
         'BEGIN'
            'FOR'I:=1'STEP'1'UNTIL'WIDTH'DO'
            PRINTCH(SHADE[RECT[I,J]]);
            NEWLINE(1);
         'END';
         PAPERTHROW;
      'END';
'END';
```

Appendix E2. The SPLT program for generating MUSIC 5 sound data.

```
'BEGIN'
   'REAL'X;
   'REAL''PROCEDURE'RNDEC(B);
      'REAL'B;
      'BEGIN' X:=9.2351*X+B;
              RNDEC:=X:=100*X-ENTIER(100*X);
      'END';
   'INTEGER''PROCEDURE'RND(N);
      'INTEGER'N;
      'BEGIN' RND:=ENTIER(RNDEC(3.573)*N)+1 'END';
   'PROCEDURE'NOTE(P,Q,R);
      'REAL'P,Q,R;
      'BEGIN'
         WRITETEXT('('NOT')');
         PRINT(P,3,4);
         WRITETEXT('('%1%')');
         PRINT(Q-P,3,4);
         PRINT(R,3,4);
         PRINT(RND(RND(8000)),4,0);
         WRITETEXT('('%%0%%0;')');
         NEWLINE(1);
      'END';
   'PROCEDURE'A(START,FINISH,INTENSITY);
      'REAL'START,FINISH,INTENSITY;
      'BEGIN'
         'SWITCH'TYPE:=MS,HS,SND,SIL;
         'GOTO'TYPE[RND(4)];
MS:      A(START,START+(FINISH-START)/2,INTENSITY);
         A(START+(FINISH-START)/2,FINISH,INTENSITY);
         'GOTO'SIL;
HS:      A(START,FINISH,INTENSITY/2);
         A(START,FINISH,INTENSITY/2);
         'GOTO'SIL;
SND:     NOTE(START,FINISH,INTENSITY);
SIL:
   'END';
   'FOR'X:=0.3,0.5,0.9'DO'
   'BEGIN'
      A(0,10,1500);
      WRITETEXT('('SEC%11;')');
      NEWLINE(1);
   'END';
  WRITETEXT('('TER%0.1')');
'END';
```

Appendix E3. MUSIC 5 sound data generated by SPLT.

```
NOT   0.0000   1   10.0000    187.5000    2747    0  0;
NOT   0.0000   1   10.0000     93.7500      16    0  0;
NOT   0.0000   1   10.0000     46.8750    3783    0  0;
NOT   0.0000   1    5.0000     23.4375    2596    0  0;
NOT   0.0000   1    5.0000      2.9297    2481    0  0;
NOT   0.0000   1    2.5000      5.8594    1928    0  0;
NOT   2.5000   1    1.2500      5.8594      27    0  0;
NOT   3.7500   1    0.3125      1.4648    2887    0  0;
NOT   3.7500   1    0.6250      1.4648    3289    0  0;
NOT   3.7500   1    0.6250      1.4648    1661    0  0;
NOT   3.7500   1    0.6250      1.4648    5319    0  0;
NOT   4.3750   1    0.3125      2.9297    1563    0  0;
NOT   4.6875   1    0.3125      0.7324    3548    0  0;
NOT   4.6875   1    0.3125      0.7324    2400    0  0;
NOT   4.6875   1    0.3125      0.7324    3238    0  0;
NOT   0.0000   1    5.0000     11.7188    1607    0  0;
NOT   0.0000   1   10.0000    375.0000    1797    0  0;
NOT   0.0000   1   10.0000    375.0000      63    0  0;
SEC 11;
NOT   0.0000   1    1.2500   1500.0000     556     0  0;
NOT   1.2500   1    0.6250    750.0000    1746    0  0;
NOT   1.8750   1    0.6250    750.0000    1049    0  0;
NOT   1.2500   1    0.0781     46.8750    3036    0  0;
NOT   1.2500   1    0.1563     46.8750    3148    0  0;
NOT   1.2500   1    0.1563     93.7500     226    0  0;
NOT   1.4844   1    0.0781     46.8750     559    0  0;
NOT   1.5625   1    0.3125     93.7500     141    0  0;
NOT   1.2500   1    0.6250    187.5000    3313    0  0;
NOT   1.8750   1    0.3125    187.5000     828    0  0;
NOT   2.0312   1    0.0391     46.8750    5018    0  0;
NOT   2.0703   1    0.0391     23.4375    4542    0  0;
NOT   2.0312   1    0.0781     23.4375    3787    0  0;
NOT   2.0312   1    0.0781     23.4375    2350    0  0;
NOT   2.0312   1    0.0781     93.7500     345    0  0;
NOT   2.1094   1    0.0781    187.5000    3701    0  0;
NOT   2.1875   1    0.3125    187.5000    3679    0  0;
NOT   1.2500   1    1.2500    375.0000     519    0  0;
NOT   2.5000   1    2.5000   1500.0000    1707     0  0;
NOT   8.7500   1    1.2500   1500.0000     384     0  0;
SEC 11;
NOT   5.0000   1    5.0000    750.0000     850    0  0;
NOT   0.0000   1    5.0000    375.0000     341    0  0;
NOT   8.7500   1    0.6250    750.0000    3799    0  0;
NOT   9.3750   1    0.1563     93.7500    1688    0  0;
NOT   9.6484   1    0.0195      5.8594    2579    0  0;
NOT   9.6484   1    0.0049      0.3662     472    0  0;
NOT   9.6533   1    0.0024      0.1831    2910    0  0;
NOT   9.6570   1    0.0006      0.1831    3838    0  0;
NOT   9.6576   1    0.0006      0.1831    1635    0  0;
NOT   9.6484   1    0.0024      0.3662    2359    0  0;
NOT   9.6509   1    0.0002      0.0916    4371    0  0;
TER 10;
```

REFERENCES

Chapter One

1. Peter Manning (1977) Electronic music and the computer: a critical study of the development of electronic music systems and the introduction of computer-based technology, with particular reference to the interface problems encountered in composer/ machine communications. Ph.D. Thesis, University of Durham.

2. for example, Stanley Haynes (1978) Computers and Synthesizers, Music and Musicians, February 1978 p.28, and a section on computer music in Alan Douglas (1973) Electronic Music Production, Pitman Publishing, pp. 109 - 119.

3. John Morehan and Ian Bent (1979) Computer Applications in Musicology, The Musical Times, July 1979, p.563.

4. a number of analysis/synthesis experiments subsequently appeared, with generally disappointing results, for example, Olson and Belar (1961) The Analysis of Stephen Foster Songs, J.Acous. Soc.Am., Vol 33 no 9 pp 1163 - 1170, this is also commented upon in Manning (1977) loc. cit..

5. Kevin Jones (1978) A practical study of the Application of Computer Techniques in Processes of Musical Composition. M.Phil Thesis, University of Aston in Birmingham.

6. Curtis Roads (1979) Grammars as Representations for Music, Computer Music Journal, Vol 3 no 1. S. R. Holtzman (1979) An Automated Digital Sound Synthesis Instrument, Computer Music Journal, Vol 3 no 2.

7. Curtis Roads (1978) Composing Grammars.

8. S. R. Holtzman (1979) GGDL, Generative Grammar Definitional Language, Dept. of Computer Science, University of Edinburgh.

Chapter Two

1. Geoffrey Leech (1974) Semantics, Penguin

2. Otto E. Laske (1975) Introduction to a Generative Theory of Music, Institute of Sonology, Utrecht, and (1978) Considering Human Memory in Designing User Interfaces for Computer Music, Computer Music Journal, Vol 2 no 4.

3. for example, Robert McMahan has reconstructed examples of late Brahms piano music reported in the article on Computer Music in New Dictionary of Twentieth Century Music, ed. Vinton, Thames and Hudson.

4. Laske op.cit..

5. Max Matthews (1969) The Technology of Computer Music, M.I.T..

6. Hubert S. Howe Jnr. (1977) Electronic Music Synthesis: Concepts, Facilities, Techniques, J.M.Dent and Sons, pp 175 - 248.

7. A MUSIC 10 tutorial, IRCAM.

8. John Chowning (1977) The Synthesis of Complex Audio Sectra by means of Frequency Modulation, Computer Music Journal, Vol 1 no 1.

9. Werner Kaegi and Stan Templaars (1978) VOSIM - A new sound synthesis system, Journal of the Audio Engineering Society, Vol 26 no 6.

10. Kees Van Prooijen (1978) CYCLE, A simple sound synthesis program, Institute of Sonology, Utrecht.

11. P. Berg (1979) PILE - A language for sound synthesis, Computer Music Journal, Vol 3 no 1

12. P. Berg (1978) A user's manual for SSP, Institute of Sonology, Utrecht.

13. S. R. Holtzman (1978) A Description of an Automated Digital Sound Synthesis Instrument, Dept. of Artificial Intelligence, Research Report no 59, University of Edinburgh.

14. Herbert Brun (1978) Dust, More Dust, Dustiny, UNESCO Workshop, Aarhus University.

Chapter Three

1. Max Matthews, op. cit., pp 78 f .

2. William Buxton (1978) Design Issues in the Foundation of a Computer Based tool for Music Composition, Technical Report CSRG-97, University of Toronto.

3. G. M. Koenig (1978) Composition Processes, UNESCO Workshop, Aarhus University.

4. Papworth (1960) Computers and Change Ringing, The Computer Journal, Vol 3 no 47

5. Lejaren Hiller (1978), lecture given at the City University, London.

6. Steve Reich (1974) Writings about Music, New York University Press.

7. Oliver Messiaen (1944/56) The technique of my musical language. See for example the "Agrandissement asymétrique" p 62 of Regard de l'Esprit de joie, X of *Vingt Regards sur l'Enfant-Jésus*, Editions DURAND, 1944.

8. Stanley Gill (1963) A Technique for the Composition of Music in a Computer, The Computer Journal, vol 6 no 2.

9. G. M. Koenig (1970) Project One, Institute of Sonology Report, Utrecht

10. B. Vercoe (1978), lecture at the UNESCO Workshop, Aarhus.

11. John Melby (1980), personal conversation with the author.

12. Donald Byrd (1977) An Integrated Music Software System, Computer Music Journal, Vol 1 no 2.

13. Edward D. Kobin and Theodore H. A. Ashford (1968) A Solution to the problems of vertical serialization, Perspectives of New Music, Spring/Summer 1968.

14. G. M. Koenig (1970), op. cit..

15. Leland Smith (1972) SCORE - A Musician's Approach to Computer Music, Journal of the Audio Engineering Society, vol 20 no 1.

16. William Buxton and Guy Fedorkow (1978) The Structured Sound Synthesis Project (SSSP), an Introduction, Technical Report CSRG-92, University of Toronto.

17. Holtzman (1979 b) , loc.cit..

18. C. Roads (1978 a) , loc.cit..

19. Iannis Xenakis (1971) Formalised Music.

20. Hiller and Isaacson (1959) Experimental Music.

21. Koenig (1970), loc.cit..

22. Barry D. Truax (1973) The Computer Composition - Sound Synthesis Programs POD4, POD5 and POD6, Institute of Sonology, Utrecht, and (1978) Computer Music Composition: The Polyphonic POD System, proc. IEEE (Computer), August 1978.

Chapter Four

1. for example, Hammersley and Handscomb (1959) Monte Carlo Methods, Methuen, and T. G. Newman and P. L. Odell The generation of Random Variates, Griffin.

2. Bela Bartok, *Microkosmos Book 6.*

3. Kevin Jones (1973) Bach and Handel, A statistical analysis of Recitative, Undergraduate Project (Appendix), University of York.

4. M. J. Moroney (1951) Facts from Figures, Penguin Books.

5. ibid., and also Truax (1973), loc.cit..

6. Xenakis (1971) loc.cit., pp 28 f .

7. Truax (1973), loc.cit..

8. Kevin Jones (1978) loc.cit..

Chapter Five

1. The Revelation of St. John, chapter 20 vs 10, Revised Standard Version.

Chapter Six

1. Lajos Takacs (1960) Stochastic Processes, Problems and Solutions, Methuen.

2. see for example, Takacs (1960) loc.cit., U.Narayan Bhat (1972) Elements of Applied Stochastic Processes, John Wiley and Sons, and V. I. Romanovsky (1970) Discrete Markov Chains, Walters-Noordhoff.

4. U. Narayan Bhat (1972), pp 87 - 96.

5. Benoit B. Mandelbrot (1977) Fractals: Form, Chance and Dimension, W. H. Freeman.

5. Richard Voss and John Clarke (1978) "I/f noise" in music: Music from I/f noise, Journal of the Acoustic Society of America, vol 60 no 1, Jan 1978, pp 258 f. , see also, Martin Gardner (1978) White and Brown Music, fractal curves and one-over-f functions, Scientific American, vol 238, April 1978, p 16.

6. see for example Debussy (1908) *The snow is dancing* from *Children's Corner* suite, United Publishers Ltd., and also *...les tierces alternées* from *Preludes Book II*.

7. P. Berg (1978) A user's manual for SSP, Institute of Sonology, Utrecht.

8. Barry Truax (1973) loc.cit..

9. Kevin Jones (1978) loc.cit..

Chapter Seven

1. Peter Stadlen (1978) Electronic Music Studio Concert, Skinner's Hal The Daily Telegraph, Friday July 21, 1978, p.13.

Chapter Eight

1. Curt Roads (1978) loc.cit..

2. S. R. Holtzman (1979) loc.cit..

3. David Robey (ed.) (1973) Structuralism, An Introduction, Clarendon Press, Oxford.

4. John Lyons (1970) Chomsky, Fontana/Collins.

5. see, K. S. Fu (1974) Syntactic Methods in Pattern Recognition, Academic Press, and Rafael C. Gonzalez and Michael G. Thomason (1978) Syntactic Pattern Recognition, An Introductio Addison-Wesley.

6. Noam Chomsky (1963) Formal Properties of Grammars, Handbook of Mathematical Psychology, vol 2, Wiley.

7. Fu (1974) loc.cit..

8. Gonzalez and Thomason (1978) loc.cit..

9. ibid. p. 189.

10. Guiseppi Verdi *Aida* (Vocal Score), Picardi, p. 140.

11. Each number in a Fibonacci sequence is the sum of the two preceding numbers: $a_n = a_{n-1} + a_{n-2}$. The simplest form of the sequence is: 1, 1, 2, 3, 5, 8, 13, 22, 34, 55, 89, ... etc..

12. Newman W. Powell (1979) **Fibonacci and the Golden Mean: Rabbits, Rumbas and Rondeaux**, Journal of Music Theory, vol 23 no 2.

13. Ernö Lendrai (1971) **Bela Bartok, An Analysis of His Music**, Kahn and Averill.

Chapter Nine

1. Xenakis loc.cit. p. 9.

2. referred to by Brian Magee (1973) in **Popper**, Fontana/Collins, see also Lero E. Loemker (1972) **Struggle for Synthesis: The Seventeenth Century Background of Leibniz's Synthesis of Order and Freedom**, Harvard University Press, which includes some interesting discussion of Leibniz's theories of universal harmony, and their relation to music.

3. Karl Popper (1972) **Objective Knowledge, An Evolutionary Approach**, Clarendon Press, Oxford, pp. 206 - 255.

BIBLIOGRAPHY

A. M. Arthurs (1965) **Probability Theory** Routledge and Kegan Paul.

Frank Ayres, Jnr. (1962) **Theory and Problems of Matrices** McGraw Hill.

Pierre Barbaud (1966) **Initiation a la Composition Musicale Automatique** Dunod.

Judith and Alton Becker (1979) **A grammar of the Musical Genre, Srepegan** Journal of Music Theory, vol 23 no 1.

Jonathan Benthall (1972) **Science and Technology in Art Today** Thames and Hudson.

P. Berg (1978) **A user's manual for SSP** Institute of Sonology, Utrecht.

P. Berg (1979) **PILE - A language for sound synthesis** Computer Music Journal vol 3 no 1.

Leonard Bernstein (1977) **The Unanswered Question** Harvard University Press.

U. Narayan Bhat (1972) **Elements of Applied Stochastic Processes** John Wiley and Sons.

Herbert Brun (1978) **Dust, more Dust, Dustiny** Lecture given at the UNESCO workshop on computer music, Aarhus University.

William Buxton (1978) **Design issues in the Foundation of a Computer Based tool for Music Composition** Computer Systems Research Group Technical Report CSRG-97, University of Toronto.

William Buxton and Guy Fedorkow (1978) **The Structured Sound Synthesis Project (SSSP), an Introduction** Technical Report CSRG-92, Computer Systems Research Group, University of Toronto.

Donald Byrd (1977) **An integrated Music Software System** Computer Music Journal, vol 1 no 2.

Noam Chomsky (1963) **Formal Properties of grammars** Handbook of Mathematical Psychology, vol 2, Wiley.

John Chowning (1977) **The Synthesis of Complex Audio Spectra by Means of Frequency Modulation** Computer Music Journal, vol 1 no 1.

John Clough (1970) **TEMPO - A composer's programming language** Perspectives of New Music, Fall/Winter 1970.

Alan Douglas (1973) **Electronic Music Production** Pitman Publishing.

Anton Ehrenzweig (1970) **The Hidden Order of Art** Paladin 1970.

Fleuret (1972) **Xenakis - A music for the future** Music and Musicians, April 1972.

Robert E. Frankel, Stanley J. Rosenchein and Stephen S. Smoliar (1976) **A LISP-based system for the Study of Schenkerian Analysis** Computers and the Humanities vol 10 no 1.

John B. Frayleigh (1967) **A First Course in Abstract Algebra** Addison Wesley.

K. S. Fu (1972) **On Syntactic Pattern Recognition and Stochastic Languages** in "Frontiers of Pattern Recognition", ed. S. Watanabe, Academic Press, New York.

K. S. Fu (1974) **Syntactic Methods in Pattern Recognition** Academic Press.

Martin Gardner (1978) **White and Brown Music, fractal curves and one-over-f functions** Scientific American vol 238, April 1978.

Stanley Gill (1963) **A Technique for the Composition of Music in a Computer** The Computer Journal vol 6 no 2.

Rafael Gonzalez and Michael Thomason (1978) **Syntactic Pattern Recognition, An Introduction** Addison Wesley.

Hammersley and Handscomb (1959)**Monte Carlo Methods** Methuen

Stanley Haynes (1978) **Computers and Synthesizers** Music and Musicians vol 26 no 6, February 1978.

Hiller and Issacson (1959) **Experimental Music** McGraw Hill

Lejaren Hiller (1970) **Music Composed with Computers - A Historical Survey** in "The Computer and Music", ed. Lincoln.

S. R. Holtzman (1978) **A Description of an Automated Digital Sound Synthesis Instrument** Department of Artificial Intelligence, Research Report no 59, University of Edinburgh.

S. R. Holtzman (1979) **An Automated Digital Synthesis Instrument** Computer Music Journal vol 3 no 2.

S. R. Holtzman (1979) GGDL, Generative Grammar Definitional Language Department of Computer Science, University of Edinburgh.

J. E. Hopcraft and J. D. Ullman (1969) Formal Languages and their Relation to Automata Addison Wesley.

Hubert S. Howe Jnr. (1975) Electronic Music Synthesis: Concepts, Facilities, Techniques J. M. Dent and Sons.

Hubert S. Howe Jnr. (1977) Electronic Music and Microcomputers Perspective of New Music vol 16 no 1.

Kevin Jones (1973) Bach and Handel, A statistical analysis of Recitative Undergraduate Project (Appendix), University of York.

Kevin Jones (1973) Automatic Methods in Musical Composition - the Computer as a Composer Undergraduate Project, University of York.

Kevin Jones (1978) A Practical Study of the Application of Computer Techniques in Processes of Musical Composition M.Phil. Thesis, University of Aston in Birmingham.

Werner Kaegi and Stan Templaars (1978) VOSIM - A New Sound Synthesis System Journal of the Audio Engineering Society, vol 26 no 6.

A. Kaufmann (1972) Points and Arrows, The Theory of Graphs Transworld Publishers.

Edward D. Kobin and Theodore H. A. Ashford (1968) A Solution to the problem of Vertical Serialization Perspectives of New Music, Spring/ Summer 1968.

G. M. Koenig (1971) The use of computer programs in Creating Music UNESCO/La Revue Musicale.

G. M. Koenig (1970) Project One Institute of Sonology Report, Utrecht State University.

G. M. Koenig (1978) Composition Processes Lecture given at the UNESCO workshop on Computer Music, Aarhus University.

Otto E. Laske (1975) Introduction fo a Generative Theory of Music Institute of Sonology, Utrecht State University.

Otto E. Laske (1978) Considering Human Memory in Designing User Interfaces for Computer Music Computer Music Journal.

Edmund Leach (1970) Levi Strauss Fontana/Collins.

Geoffrey Leech (1974) Semantics Penguin.

Ernö Lendvai (1971) Bela Bartok, An Analysis of His Music Khan and Averill, London.

Lincoln (1970) The Computer and Music Cornell University Press.

Lincoln (1970) Uses of the Computer in Music Composition and Research Advances in Computers vol 12, Academic Press.

Lero E. Loemker (1972) Struggle for Synthesis: the Seventeenth Century Background of Leibniz's Synthesis of Order and Freedom Harvard University Press.

J. D. Lomax (ed.) (1973) Computers in the Creative Arts The National Computing Centre Limited.

H. C. Longuet-Higgins and M. J. Steadman (1971) On Interpreting Bach Machine Intelligence vol 6 no 15, Edinburgh University Press.

John Lyons (1970) Chomsky Fontana/Collins.

Bryan Magee (1973) Popper Fontana/Collins.

Benoit B. Mandlebrot (1977) Fractals: Form, Chance and Dimension W. H. Freeman and Company, San Francisco.

Peter Manning (1977) Electronic Music and the Computer Ph.D. Thesis, University of Durham.

Max Matthews (1969) The Technology of Computer Music Massachusetts Institute of Technology.

Oliver Messiaen (1944/56) The Technique of my musical language Alphonse Leduc (trans. John Satterfield).

Leonard B. Meyer (1967) Music the Arts and Ideas University of Chicago Press.

George A. Miller (1967) The Psychology of Communication Penguin.

John Moreham and Ian Bent (1979) Computer Applications in Musicology The Musical Times, July 1979, p 563.

M. J. Moroney (1951) Facts from Figures Penguin Books.

Robert E. Mueller (1967) The Science of Art, The Cybernetics of Creative Communication Rapp and Whiting

T. G. Newman and P. L. Odell The generation of Random Variates Griffin.

J. D. O'Connor (1973) Phonetics Penguin.

Olson and Belar (1961) The Analysis of Stephen Foster Songs Journal of the Acoustical Society of America, vol 33 no 9, pp 1163 - 1170.

Papworth (1960) Computers and Change Ringing The Computer Journal vol 3 no 47.

Pierce (1962) Symbols, Signals and Noise Hutchinson 1962.

Pierce (1971) A Chance for Art in "Cybernetics, Art and Ideas", ed. Reichardt.

Karl Popper (1962) The Open Society and its Enemies Routledge and Kegan Paul.

Karl Popper (1972) Objective Knowledge, An Evolutionary Approach Clarendon Press, Oxford.

Newman W. Powell (1979) Fibonacci and the Golden Mean: Rabbits, Rumbas and Rondeaux Journal of Music Theory vol 23 no 2.

Steve Reich (1974) Writings about Music New York University Press.

Jasia Reichardt (ed.) (1968) Cybernetic Serendipity Studio International.

Jasia Reichardt (1971) The Computer in Art Studio Vista.

Jasia Reichardt (1978) Cybernetics, Art and Ideas Studio Vista.

C. Roads (1978) Automatic Granular Synthesis of Sound Computer Music Journal vol 2 no 2.

C. Roads (1978) Composing Grammars Unpublished version of a paper presented at the 1977 International Computer Music Conference, University of California, San Diego in La Jolla.

C. Roads (1978) An Interview with Gottfried Michael Koenig Computer Music Journal vol 2 no 3.

C. Roads (1979) Grammars as Representations for Music Computer Music Journal vol 3 no 1.

David Robey (ed.) (1973) Structuralism, An Introduction Clarendon Press, Oxford.

V. I. Romanovsky (1970) Discrete Markov Chains Walters Noordhoff Publishing, Groningen.

W. Ross Ashby (1956) An Introduction to Cybernetics Chapman and Hall.

Pierre Schaeffer (1971) La Musique et les Ordinateurs UNESCO/La Revue Musicale.

Robert Sherlaw Johnson (1975) Messiaen J. M. Dent and Sons, London.

Reginald Smith Brindle (1975) The New Music Oxford University Press.

Stephen S. Smoliar (1976) **Music Programs: An Approach to Music Theory through Computational Linguistics** Journal of Music Theory vol 20 no

Peter Stadlen (1978) **Electronic Music Studio Concert, Skinner's Hall** The Daily Telegraph, Friday July 21, 1978.

Peter S. Stevens (1974) **Patterns in Nature** Penguin Books.

Harold S. Stone (1973) **Discrete Mathematical Structures and their Applications** Science Research Associates.

Anthony Storr (1972) **The Dynamics of Creation** Penguin.

Stuckenschmidt (1969) **Twentieth Century Music** Weidenfeld and Nicholson.

Alan Sutcliffe (1973) **Examples of the Use of Computers in Music** in " Computers in the Creative Arts ", ed. Lomax.

Lajos Takacs (1960) **Stochastic Processes, Problems and Solutions** Methuen.

C. A. Taylor (1965) **The Physics of Musical Sounds** The English Universities Press.

Barry D. Truax (1973) **The Computer Composition - Sound Synthesis Programs POD4, POD5 and POD6** Institute of Sonology, Utrecht State University.

Barry D. Truax (1977) **A Communicational Approach To Computer Sound Programs** Journal of Music Theory vol 20 no 1.

Barry D. Truax (1978) **Computer Music Composition: The Polyphonic POD System** proc IEEE (Computer), August 1978.

Kees Van Prooijen (1978) **CYCLE, A simple sound synthesis program** Institute of Sonology, Utrecht State University.

Von Foerster and Beachamp (1969) **Music by Computers** John Wiley.

Richard Voss and John Clarke (1978) **"I/f noise" in music: Music from I/f noise** Journal of the Acoustic Society of America vol 63 no 1, Jan 1978, p 258.

Iannis Xenakis (1971) **Formalized Music, Thought and Mathematics in Composition** Indiana University Press.

Zaripov (1969) **Cybernetics and Music** Perspectives of New Music vol 7 no 2.

INDEX