

CHAPTER 5 — *Jam Factory* — *Étude in Memoriam Allan Dagg*

5.1. Overview

The *Étude in Memoriam Allan Dagg* (Track 2 on the accompanying audio CD) was initiated following the death of Allan Dagg, an amateur swing and Dixieland jazz trumpet player and professional carpenter. The work consists of six separate interactions between composer and computer, each interaction forming a separate section of the work. The six sections are linked with MIDI sequencing software to form a single movement. As a memorial to Allan Dagg, the work consists of trumpet samples and samples of carpentry sounds, while FM generated bell timbres relate to his death. The overall form of the work is governed by the timbres used, the six sections each developed around the use of different timbres or timbral combinations. The sections are grouped into pairs to form three main parts: Part 1 focuses on FM bell timbres; Part 2 on trumpet samples; and Part 3 on carpentry samples.

The *Étude in Memoriam Allan Dagg* is a programmatically-based work, the first sections in Parts 1-3 programmatically portraying personal memories of Allan Dagg's life and work. The first portrayal relates to his death, through the tolling of funeral bells. The second portrayal, using trumpet samples, is of his sonic repertoire in relation to his life as a jazz trumpeter. The third portrayal relates to his life as a carpenter, concentrating on sounds from his carpentry trade. Following each of these portrayals are three further programmatic portrayals that form the second section in each of the three parts of the work, and build upon their antecedents: the first, following the bells, is a representation of an after-life or place after death, the second, building on trumpet samples, represents a last chorus for the jazz trumpeter, and the third, as a textural and programmatic climax within the work, represents a personal farewell.

Jam Factory, the software used for the development of the *Étude* interactions, is described by its authors as an 'improvisation and live performance processor' (Chadabe & Zicarelli 1986a). The real-time, improvisation style capabilities of the program form a basis for the development of the *Étude*. Pitch and rhythmic materials stored in the design stage of each section are subject to Markov processes in the operation stages of the program,

processes that result in improvisation style variations on the stored pitch and rhythmic materials. The following sections describe the musical materials used in the *Étude*, prior to algorithm examinations that include a detailed examination of Markov processes.

5.2. Composition Structures

5.2.1. Pitch structures

Pitch structures in the *Étude in Memoriam Allan Dagg* are entirely based on the opening eight measures of the jazz standard *All of Me* by Seymour Simons and Gerald Marks (Simons & Marks 1931). This excerpt, as shown in Figure 5.1, was initially chosen as a basis for the *Étude* due to its diverse range of rhythmic durations and pitch, the eight excerpt measures containing nine separate rhythmic durations, and 10 pitches from the total chromatic scale.

Figure 5.1 *All of Me*, mm.1-8



Pitch structures in the opening section of the *Étude* are based on divisions of the original melody into the three fragments shown in Figure 5.2. Three *Jam Factory* voices or *players* are assigned a single fragment, while a fourth player is assigned a repeated C, the repetition forming an ostinato bell tone that opens the work and continues throughout the section. Similar to *M*'s representation of stored pitch, *Jam Factory* represents stored pitch

with a piano scroll and grid, Figure 5.3 showing the scroll and grid for the *Jam Factory* player containing fragment 1.

Figure 5.2 *Étude in Memoriam*, Section 1, *All of Me* fragmentation

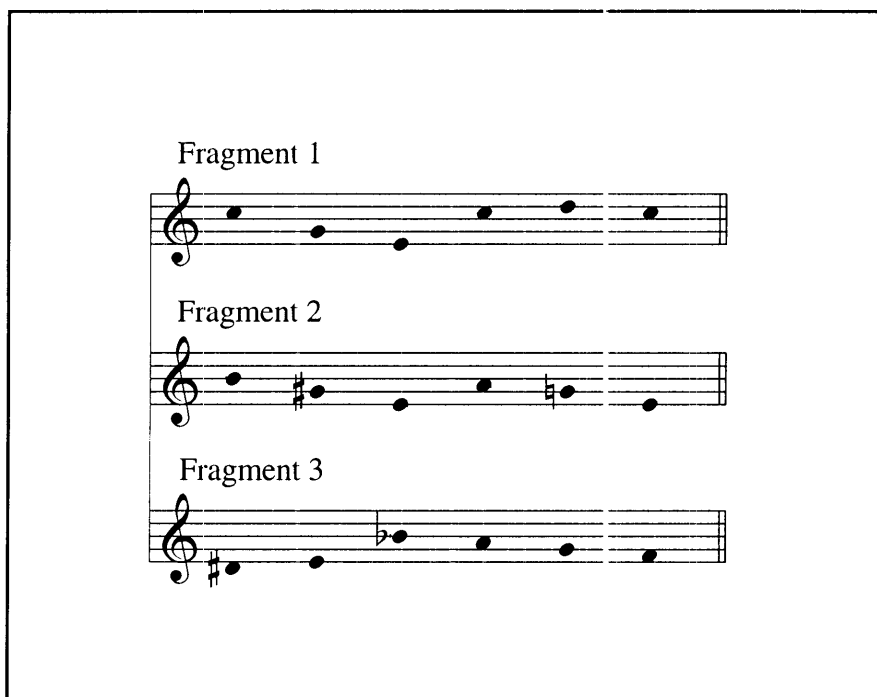
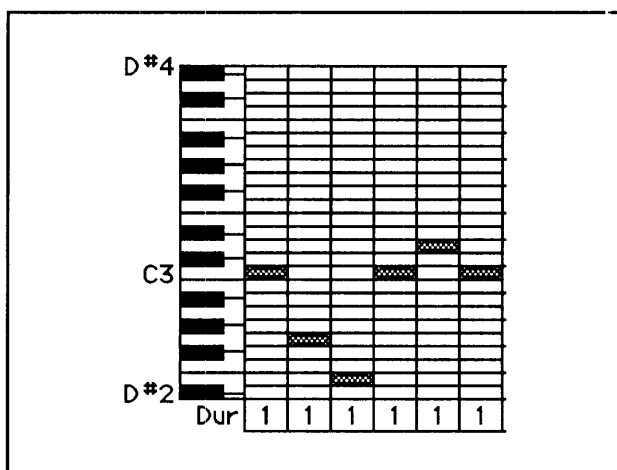
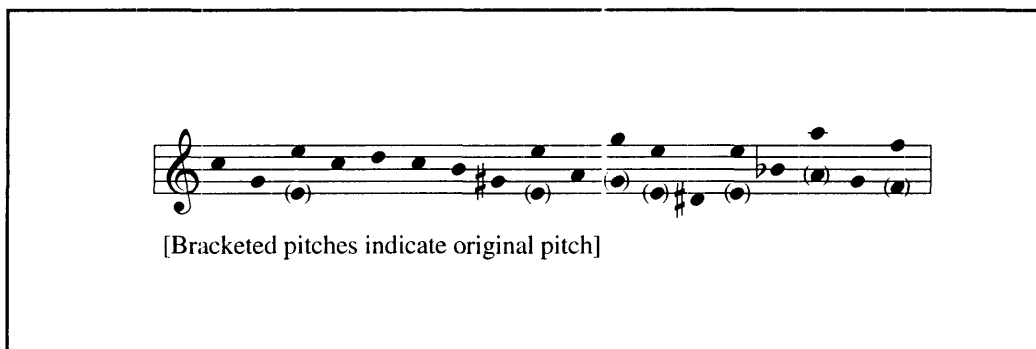


Figure 5.3 *Étude in Memoriam*, piano scroll, fragment 1



various lower frequency pitches within the excerpt forming a stereotypical reference to the after-life with which the section is programmatically concerned. This pitch series is assigned to two *Jam Factory* players, while a third player is assigned a repetitive C bell tone, similar to that used in the opening section.

Figure 5.4 *Étude in Memoriam*, Section 2 pitch material



durations, and newly-composed rhythmic duration series. Each *Jam Factory* player is controlled with an algorithm similar to *M's* Time-base algorithm, numerator/denominator settings providing a duration unit for a player. Beneath each pitch represented in the *Jam Factory* piano scroll is a numerical duration setting as shown in Figure 5.3. Where durations are all set to a value of 1, as in Figure 5.3, single value repeated rhythmic durations occur. Increments in these settings result in rhythmic series with varied durations, or in series that can be matched to durations in *All of Me*.

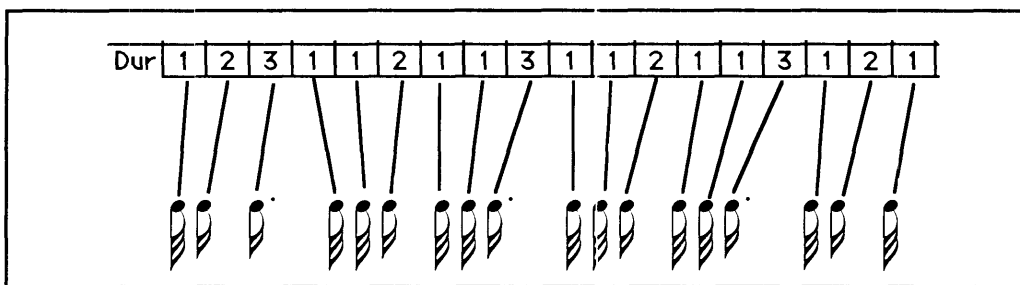
Table 5.1 shows the Time-base settings for the opening section of the piece. In all players, the piano-scroll durations are set to a value of 1, resulting in the repetition of the singular values set in the Time-base. Player 4 in this section is assigned the repeated ostinato bell tone, the remaining players are assigned normal bell tones.

Table 5.1 *Étude in Memoriam*, Part 1, Section 1, durations

Player	Time-base	Duration
1	1 / 1	
2	1 / 4	
3	1 / 3	
4	2 / 1	

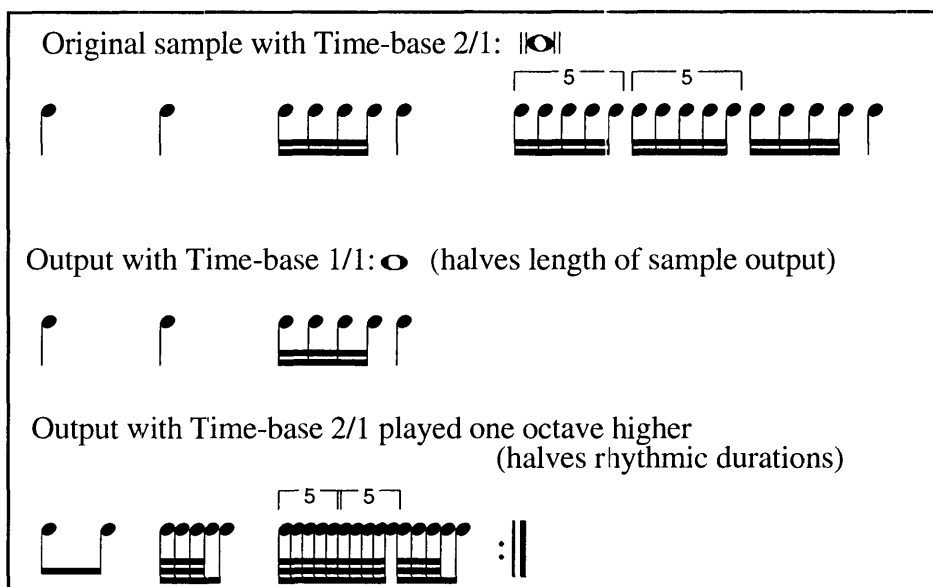
The second section of Part 1 introduces a newly-composed duration sequence for Player 2, the remaining two players in the section employing singular repeated durations: Player 1 receives a setting of 3/1, Player 3, with the repeated C bell tone, receives a setting of 6/1. Figure 5.6 shows the newly-composed sequence for Player 2 and the resulting durations with the player's 1/32 Time-base setting. Within the section, Player 2 is assigned an altered bell timbre that is reduced to the attack-only portion of an original bell timbre.

Figure 5.6 *Étude in Memoriam*, Part 1, Section 2, durations, Player 2



Part 2 of the *Étude* employs single value repeated rhythmic durations throughout, a differing set of durations in each of its two sections. Notable in Part 2 is the effect of a trumpet sample on the rhythmic output of a player employing the sample. The original sample is a recording of rhythmically varied trumpet double-tonguings, triggered in *Jam Factory* by a single rhythmic duration. The duration setting controls the length of the sample, while tessitura placements, affecting the length of the waveform, contribute to the sample's rhythmic output. Figure 5.7 shows an approximate transcription of the original sample rhythm, the effect of halving the Time-base setting, and the effect of a tessitura placement one octave higher.

Figure 5.7 *Étude in Memoriam*, Part 2, double-tongued trumpet sample durations

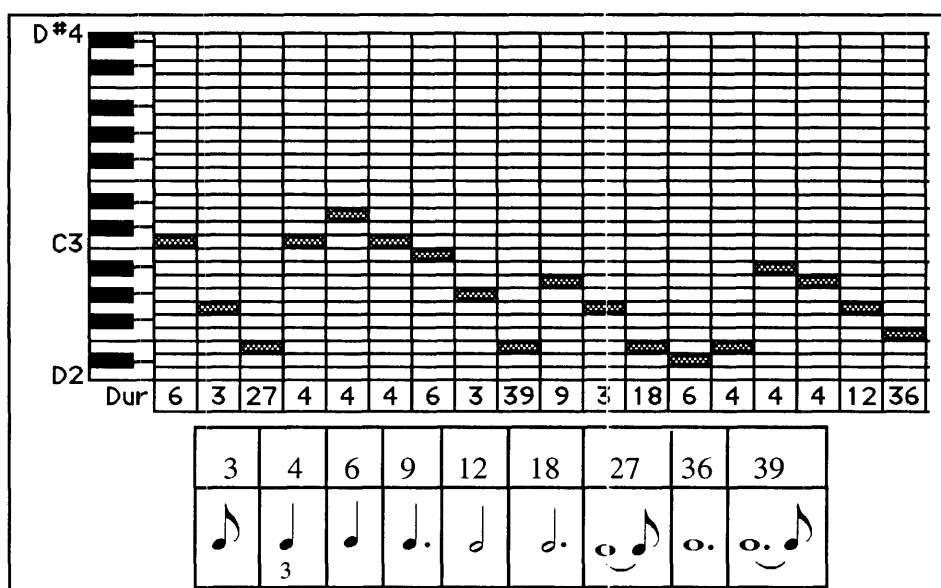


Rhythmic structures in Part 3 of the *Étude* are a combination of a newly-composed duration sequence and a duration sequence from the *All of Me* excerpt. Both sections in Part

3 employ identical Time-base settings. The newly-composed sequence is assigned to Player 1 and is in combination with the three-pitch fragment shown in Figure 5.5. The first pitch is assigned a value of 4, and the remaining pitches are assigned a value of 1 (4 1 1). Similar to the double-tongued trumpet sample in Part 2, the timbre assigned to this player (a carpenter's saw) has multiple rhythmic values within the sample: approximately ♪♪♪♪♪♪♪♪, each 1/2 note value representing one saw stroke. The Time-base setting of the player is 4/1, corresponding to the length of the original sample recorded at C3. The higher pitches in the fragment result in smaller rhythmic values due to a shortening of the waveform, while the duration setting of 1 reduces the length of the sample to its first quarter. As an example, the G3 pitch from the fragment will produce a shortened waveform with rhythmic values approximately half those in the original sample: ♪♪♪♪♪♪♪♪, while the duration setting value of 1 reduces the length of this waveform to its first quarter: ♪♪♪.

The remaining three players in Part 3 of the *Étude* are all assigned durations from the *All of Me* excerpt. In order to reproduce the excerpt's durations, a value of 3 is used to correspond to a 1/8th note rhythmic duration. Figure 5.8 shows the piano scroll corresponding to the excerpt's pitch and duration, the table below the scroll correlating the numeric and common notation symbols for the excerpt's rhythmic durations.

Figure 5.8 *Étude in Memoriam*, Part 3, durations, Players 2-4



Various Time-base settings for the players assigned the *All of Me* duration series result in a polyrhythmic texture within Part 3. As shown in Figure 5.8a, each player has a different Time-base setting that results in a different length of the original series. Player 2 proceeds in 1/32nd note values: 6 x 1/32, 3 x 1/32, 27 x 1/32 etc. Player 3 proceeds in 1/8th note triplet values, Player 4 proceeds in 1/2 note values.

Figure 5.8a *Étude in Memoriam*, Part 3, common notation, Players 2-4

The figure displays three staves of musical notation in 4/4 time, illustrating polyrhythmic textures for Players 2, 3, and 4. The notation is divided into three measures by vertical bar lines.

- Player 2 — Time-base : 1/32**: The first measure contains six eighth notes. The second measure contains three eighth notes followed by a quarter note. The third measure contains a half note.
- Player 3 — Time-base: 1/18**: The first measure contains a quarter note, an eighth note, and a quarter note. The second measure contains a quarter note, an eighth note, and a quarter note. The third measure contains a quarter note, an eighth note, and a quarter note. Above the first two measures are brackets labeled "18:16". Above the third measure is a bracket labeled "18:16". To the right of the third measure is a bracket labeled "18".
- Player 4 — Time-base: 1/2**: The first measure contains a half note. The second measure contains a half note. The third measure contains a half note.

5.2.3. Timbre

Timbres employed within the *Étude in Memoriam Allan Dagg* are sourced from three hardware devices: an FM synthesiser (Yamaha DX100), an FM sound module (Yamaha FB01) and a sampler (Akai S-950). The FM devices provide bell timbres used throughout the work, while the sampler provides various trumpet timbres and timbres related to the carpentry trade.

Timbres provided by the FM synthesiser are a normal bell tone, and two edited timbres derived from that original tone. Editing of FM timbres, via the manipulation of each patch's MIDI parameters, is carried out using *MIDIQuest*, a software program for storing and editing MIDI parameters that affect the FM synthesiser's or FM sound module's synthesised sound. The first edited timbre is a bell tone with the attack and decay portions of the tone reduced to a minimum. The reduction of the attack portion results in the lack of a characteristic bell strike, while the reduced decay portion results in a sustaining of the timbre. The second edited timbre is almost identical to the first but with the addition of

glissandi throughout the sustain. Timbres provided by the FM module are a normal bell tone and an edited version of the same tone. The edited timbre is a truncated bell tone, reduced to the attack portion of the sound, which results in a loss of the characteristic sustain and decay of normal bells.

Five trumpet samples are provided by the sampler within the *Étude*. The samples are a normal trumpet sound, a trumpet with wa-wa mute, a ‘tongued’ trumpet sample, a ‘scream’ through the trumpet, and a normal trumpet sound starting with short *glissandi* moving from below and above pitch to unison. Two carpentry-based samples are used within the work. A hammered nail provides the first, and is used in mono-tone, resembling the normal hammering of nails, and also melodically, dissociating the sound from its normal setting. The second sample is the wood saw. Table 5.2 provides a comprehensive list of timbres used in the *Étude*, the bullets within the table indicating the parts and sections in which individual timbres appear.

Table 5.2 *Étude in Memoriam*, Timbre

FM Synth.	Part 1 Section 1	Part 1 Section 2	Part 2 Section 1	Part 2 Section 2	Part 3 Section 1	Part 3 Section 2
Normal Bell tone	•					
Sustained bell tone		•				
Sustained bell w/glissandi			•	•		•
FM Module						
Normal Bell tone	•	•		•		•
Bell tone truncated		•	•		•	•
Trumpet Samples						
Normal Trumpet			•			
Trumpet w/ gliss attack				•	•	•
Trumpet w/ wa-wa mute			•	•		
Trumpet tongued			•	•		
Trumpet scream				•		

Table 5.2 (cont.)

Carpentry Samples							
Nail mono-tone						•	•
Nail pitched						•	•
Saw						•	•

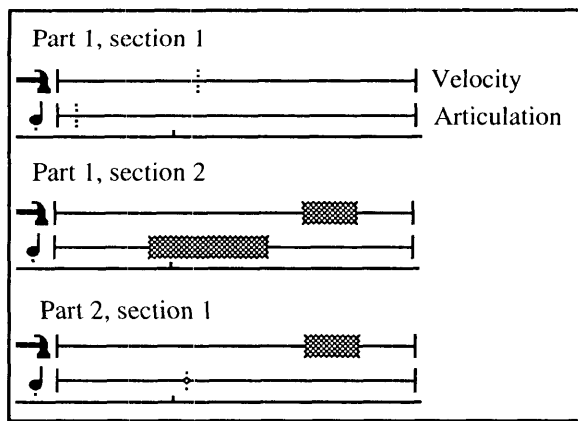
5.3. The Application of Algorithms

5.3.1. Deterministic algorithms

Purely deterministic algorithms within *Jam Factory* are limited to a Transposition algorithm and a Volume-control algorithm. The Transposition algorithm is similar to *M's* Transposition algorithm, providing adjustments to pitch by semitones and octaves. Throughout the *Étude*, transpositions are initially limited to octave adjustments, retaining the original pitch structures derived from the *All of Me* excerpt. The Volume-control algorithm allows the setting of the master volume for each *Jam Factory* player independently, while further adjustments in player dynamics are controlled with the Velocity-range algorithm described in section 2.2.3. Within the *Étude*, the Transposition and Volume-control algorithms are used extensively in the operation stage of *Jam Factory*, providing real-time alterations to transposition and volume, and are further discussed in section 5.3.3.

Jam Factory algorithms that can be set either deterministically or stochastically are the Velocity-range and Articulation-range algorithms. Unlike the *Study for Triangles*, where only deterministic settings are used for such algorithms, the *Étude* makes use of deterministic and stochastic settings. As an example, Figure 5.9 shows the velocity and articulation settings for Player 1 in the first three sections of the *Étude*. The first section has a deterministic setting in both the velocity range and articulation range, the second section uses stochastic settings to give various velocity and articulation values, and the third section has a stochastic setting for velocity, and a deterministic setting for articulation.

Figure 5.9 *Étude in Memoriam*, Velocity/Articulation settings, Part 1 and Part 2, Section 1



A deterministic algorithm for providing further control over rhythm in *Jam Factory* is its Time-distortion algorithm. This algorithm is ‘used to give players their own idiosyncratic feel when used subtly, or to create jerky timing effects when used in an extreme way’ (Chadabe & Zicarelli 1986a, p.12-1). Within MIDI software, time signature denominators specify a number of internal clock ‘ticks’ per beat, the 1/4 note value usually representing 96 ticks. Time-distortion algorithm settings, as shown in Figure 5.10, are represented with an x y axis graph, the diagonal line from lower left to upper right indicating how time progresses over a specified period of clock ticks:

At each clock tick, *Jam Factory* may either do “more” or “less” than it would normally do. If it does more, the music will sound faster. If it does less, the music will sound slower. However, at the end of the specified period, everything will have caught up to where it would normally be (Chadabe & Zicarelli 1986a, p.12-1).

The ‘subtle’ Time-distortion algorithm map in Figure 5.10 is used by voice 1 in the opening section of the *Étude*, and has a specified period of two whole notes (768 clock ticks). The diagonal line indicates an initial decrease or *rallentando* (over the first 1/2 note of the period), before a gradual increase or *accelerando* over the remainder of the time period that returns the output to its original tempo at the end of the period. The effect of the map on Player 1’s rhythm (repeated whole notes) is shown in Figure 5.11, two periods of the time map covered in the example.

Figure 5.10 *Étude in Memoriam*, Time-distortion algorithm map, Part 1, section 1

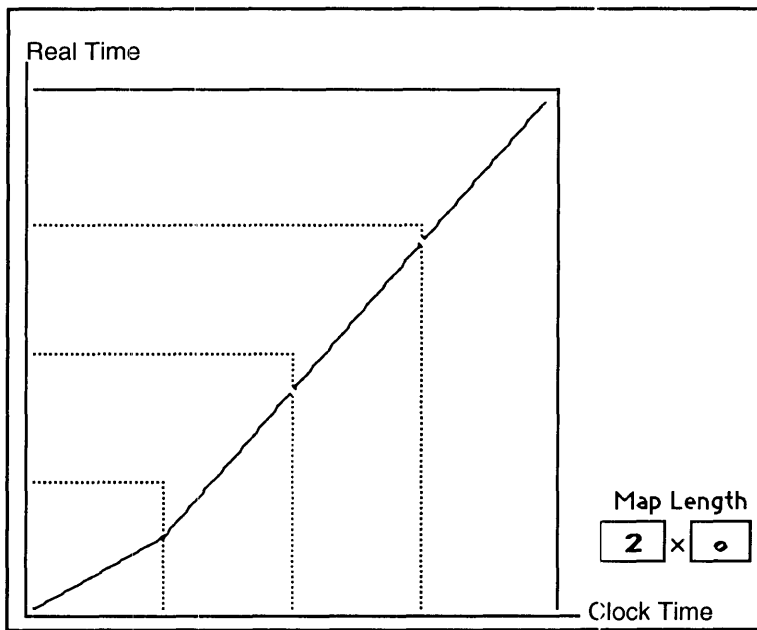
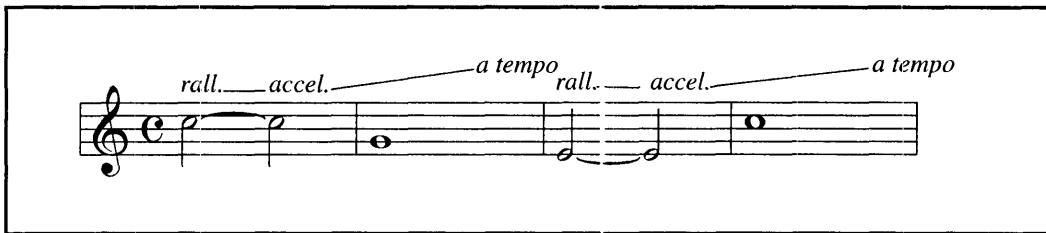
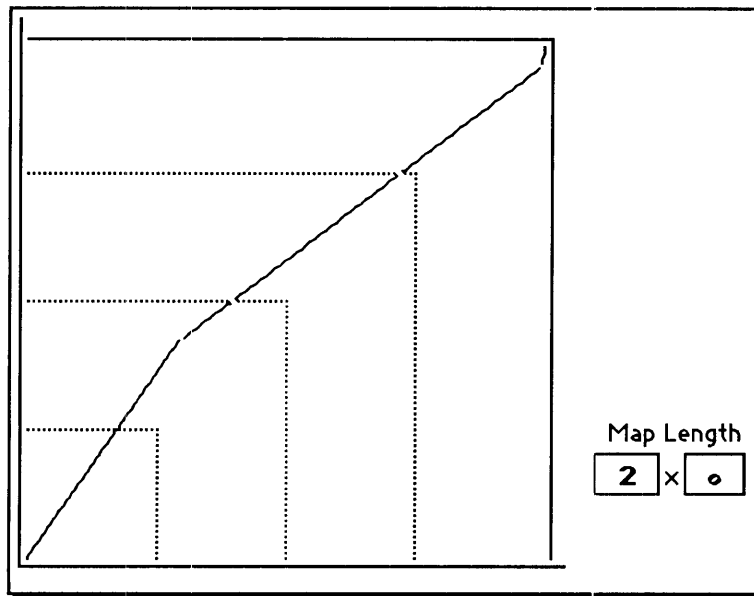


Figure 5.11 *Étude in Memoriam*, Time-distortion algorithm effect Part 1, section 1



A second Time-distortion algorithm map, shown in Figure 5.12, is used in the opening section of the *Étude* and in contrast to the first map, provides an initial *accelerando* before slowing to its original tempo. In the context of timbre in the opening section, both Time-distortion algorithm maps contribute to a depiction of the quasi-random tolling of actual cathedral bells.

Figure 5.12 *Étude in Memoriam*, Time-distortion algorithm map #2, Part 1, section 1



Similar Time-distortion algorithm maps are employed in the remaining 2 parts of the *Étude*. The maps provide subtle tempo deviations to material presented with the truncated bell tone in Parts 1 and 2, the normal trumpet timbres in Part 2, and the mono-tone nail timbre in Part 3.

5.3.2. Stochastic algorithms

The two primary stochastic algorithms within *Jam Factory* are the Silences algorithm, and the Transition table algorithm in which Markov processes are implemented. The Silences algorithm implements the same process as the Note-density algorithm within *M*, drawing on a biased choice process to control percentages of notes and silences, while transition tables provide improvisation style orderings of pitch and duration. In the operation stage of *Jam Factory* the user may choose to use either the Silences algorithm or transition tables. Throughout the *Étude in Memoriam Allan Dagg*, transition tables are used, the improvisation style orderings of pitch and duration from the tables being congruent with the improvisation-based jazz styles performed by Allan Dagg, and the jazz standard *All of Me*.

Markov processes were first formulated in 1906, the Russian mathematician Andrei Andreiëvich Markov initially using his model to ‘distill tendencies of spelling in written

Russian from the text of Aleksandr Pushkin's *Eugene Onegin*' (Ames 1988, p.175). As previously noted, Markov processes have been applied in a wide variety of fields and:

are a primary topic of study in the mathematical discipline of stochastic processes, where they have been used to model random processes, such as the changing fortunes of a gambler, the inventory of a commodity as it is affected by continuing pressures from supply and demand, or genetic fluctuations under random matings and mutations (Ames 1988, p.176).

Musical applications of Markov processes were initially based on Claude Shannon's Information Theory, music analysts, for example Harry Olson (Dodge & Jerse 1985, pp.285-8) and Richard Pinkerton (Pinkerton 1956, pp.77-86), recognising that both Markov processes and Information Theory were 'potential tools for distilling the norms and deviations of musical styles' (Ames 1988, p.176). Composition applications of Markov processes initially occurred in the late 1950s, in Hiller and Isaacson's 1957 *Illiac Suite*, and in Xenakis' 1959 works *Analogique A*, *Analogique B* and *Syrmos* (Matossian 1986, pp.134-6). While Markov processes have been used a number of times since the 1950s, they have gained little enthusiasm from composers who do not rely on automated decision-making. The processes have, however, been employed recurrently in composition programs over the past three decades (Ames 1988, pp.175-6), and are incorporated in commercial software programs such as: Charles Ames' *Compose* (Ames 1992, pp.68-71); Phil Burk, Larry Polansky and David Rosenboom's *HMSL* (Polansky, Burk & Rosenboom 1990, p.166); *Jam Factory* (Zicarelli 1987, p.24); and as an object for *MAX* by Orm Finnendahl (Nelson 1995, p.1).

The biased choice process previously described in Chapter Four is equivalent to a Markov process called a *zeroth-order Markov process*, however 'the idea of a Markov process also includes the element of *statistical analysis* of a set of data' (Moore 1990, p.430). In the opening eight measures of *All of Me* there are 18 pitch occurrences, each characterised statistically using probability percentages. For example, the pitch C occurs thrice within the excerpt resulting in a probability of 3 in 18 pitch statements, or 16.5%. A table of pitches and their corresponding probabilities is then drawn, as shown in Table 5.3. Using the table as a starting point, a zeroth-order variation of the original melody can be derived from the probabilities as shown in Figure 5.13. All of the original pitches still occur

in a zeroth-order variation, for example the three original C pitches occur in Figure 5.13, however their linear placement in the overall pitch sequence is variable.

Table 5.3 *All of Me*, Statistical analysis

Pitch	Probability %
C	16.5
D	5.5
D#	5.5
E	22.0
F	5.5
G	16.5
G#	5.5
A	11.0
Bb	5.5
B	5.5

Figure 5.13 *All of Me*, Zeroth-order Markov variation



Zeroth-order Markov variations represent simple random re-orderings in which the overall probability distribution of events in the original material is maintained. Events may be pitches, durations, timbres, dynamics or any other musical parameter from which a probability table can be derived. Within zeroth-order process, successive events are chosen independently of preceding and succeeding events, resulting in variations such as those in Figure 5.13 that have little resemblance to the original.

Further Markov processes provide variations that more closely resemble original materials by taking into account transitions from one event to the next (*1st-order Markov chains*), or from a series of events to the next event (*nth-order Markov chains*). In a 1st-order Markov chain, if an event e is always followed by another event e_1 then there is a 100% probability that e_1 will occur after e . If e is followed by e_1 and later in the event

sequence by e_2 then there is a 50% probability of e_1 occurring after e , and a 50% probability of e_2 occurring after e . As an example, the C pitch (e) in the *All of Me* excerpt, occurs thrice, followed firstly by a G (e_1), secondly by a D (e_2) and finally by a B (e_3). In a 1st-order Markov variation, as there are three alternative pitches following C, each alternative has a 33% probability of occurrence. Using these figures, a row of probabilities for pitches following C can be drawn, as shown in Figure 5.14.

Figure 5.14 *All of Me*, pitch probabilities following C

		Next Event		
		G	D	B
Current Event	C	33%	33%	33%
		Probability of occurrence		

Using this row as a basis, a matrix of probabilities, as shown in Figure 5.15, can be drawn to include each pitch in the *All of Me* excerpt. Within the matrix, percentages are, to follow standard practice, replaced with decimals, and where there is a 0% probability of one pitch following another, the corresponding space in the matrix is left blank.

Figure 5.15 *All of Me*, 1st-order pitch probability matrix

		Next Event									
		C	D	D#	E	F	G	G#	A	Bb	B
Current Event	C		0.33				0.33				0.33
	D	1.00									
	D#				1.00						
	E	0.25		0.25					0.25	0.25	
	F	1.00									
	G				0.67	0.33					
	G#				1.00						
	A						1.00				
	Bb								1.00		
	B							1.00			

Markov matrices, because they take into account the relative motions or transitions between events, are known as *transition tables*. Using the transition table in Figure 5.15, new pitch sequences can be generated to produce variations on the original sequence. Figure 5.16 shows a possible variation, the 100% probabilities in the matrix leading to numerous short pitch sequences or motifs that are identical to short motifs in the original version. Notable motifs are indicated with brackets in Figure 5.16.

Figure 5.16 *All of Me*, 1st-order variation



Figure 5.17 *All of Me*, 2nd-order pitch probability matrix

		Next Pitch									
		C	D	D#	E	F	G	G#	A	Bb	B
Current Event	C-G				1.0						
	G-E	0.5		0.5							
	E-C		1.0								
	C-D	1.0									
	D-C										1.0
	C-B							1.0			
	B-G#				1.0						
	G#-E								1.0		
	E-A						1.0				
	A-G					0.5	0.5				
	E-D#					1.0					
	D#-E									1.0	
	E-Bb								1.0		
	Bb-A						1.0				
	G-F	1.0									
	F-C						1.0				

Figure 5.18 *All of Me*, 2nd-order variation

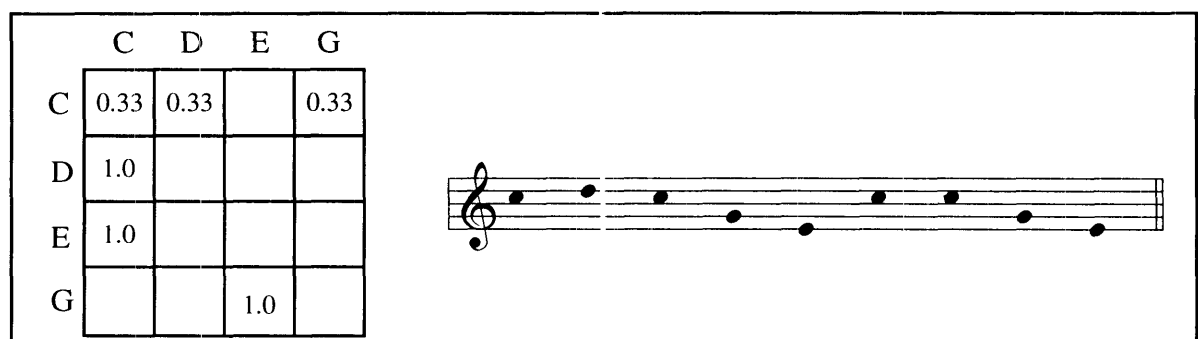


Notable in the Figure 5.17 transition table is a lack of decimals lower than 0.5. To achieve decimal values of 0.5 and lower in a 2nd-order chain, there must be more than one occurrence of a two-pitch event, each occurrence followed by a different pitch. In the above transition table this occurs twice only, with the current events G-E and A-G. As shown by the brackets in Figure 5.18, this results in a variation that again has identical pitch motifs to

those in the original, but are longer than those generated by a 1st-order transition table. Higher order transition tables, 3rd-order, 4th-order etc., generally result in variations that move closer toward original material. In the case of the *All of Me* excerpt, 3rd-order and higher transition tables will reproduce the original excerpt without variation.

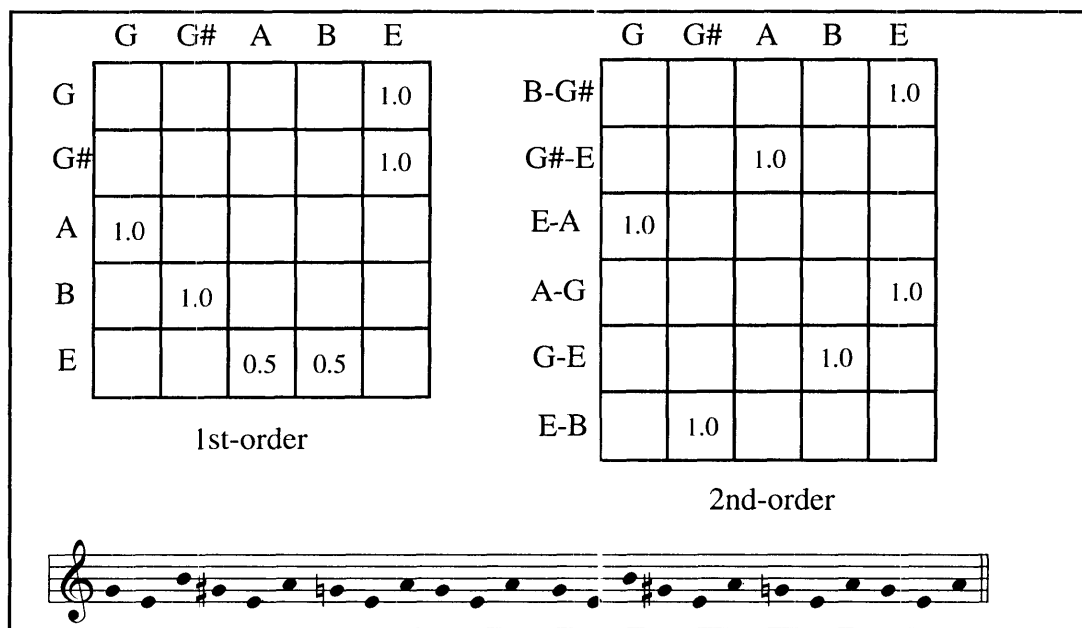
Figure 5.19 *Jam Factory*, Mixed transition table probabilities

In the opening section of the *Étude*, Markov processes are applied to all three pitch fragments from the *All of Me* excerpt. 1st-order processes are assigned to the first fragment as shown in Figure 5.20, a possible output in common notation also given in the Figure.



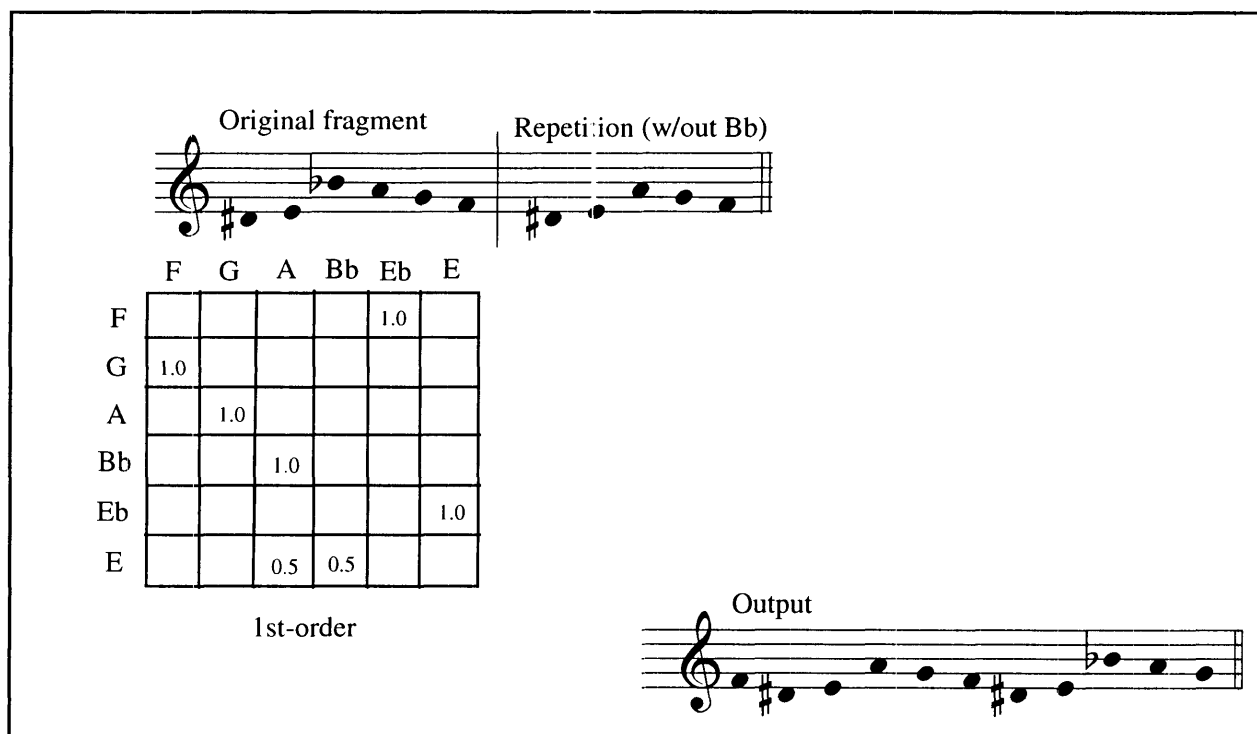
The second fragment uses the biased choice mixing algorithm and is subject to a 1st-order table 75% of the time and a 2nd-order table 25% of the time. The combined 1st-order and 2nd-order tables result in slight variations on the original material, the (25%) 2nd-order table interrupting variations produced by the (75%) 1st-order table. Figure 5.21 shows the two transition tables for the second fragment. A notated version resulting from the combined 1st-order and 2nd-order transition tables is also given in the Figure. Wherever the pitch sequence G-E occurs, the 2nd-order table will generate a B, while the 1st-order table, with E as the current event, will generate either a B (50%) or an A (50%).

Figure 5.21 *Étude in Memoriam*, Pitch fragment 2, transition tables and output



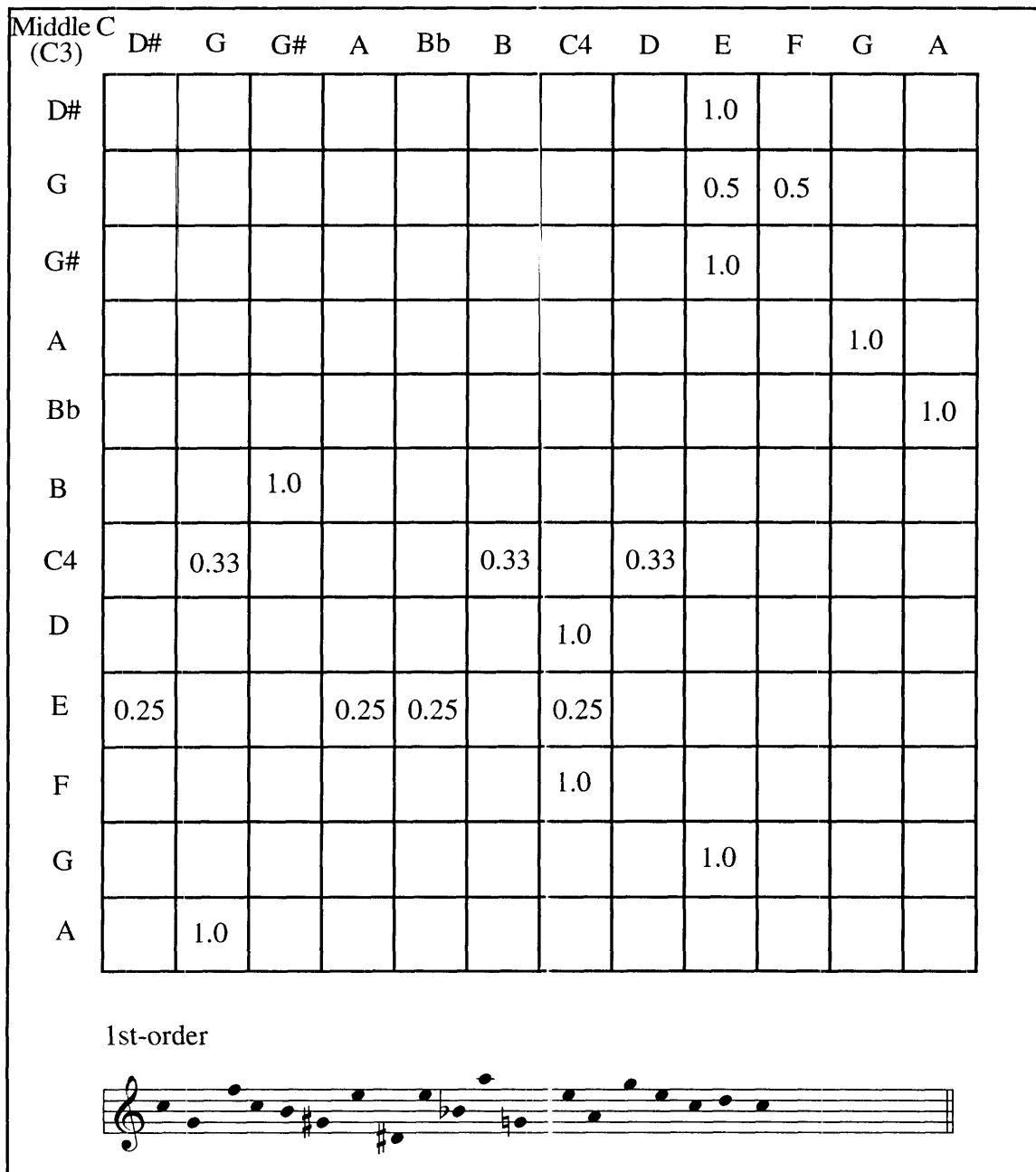
The lack of repeated pitches in the third fragment renders the application of any transition table ineffectual. However, throughout the opening section of the *Étude*, the fragment is repeated, the repetition of the fragment, as shown in Figure 5.22, excluding the Bb pitch. The alternate exclusion of the Bb pitch results overall in a 50% probability of Bb following E (from the original statements) and a 50% probability of A following E (from alternate statements), allowing an application of a 1st-order transition table.

Figure 5.22 *Étude in Memoriam*, Pitch fragment 3, transition tables and output



The second section of the *Étude* employs a 1st-order transition table with the complete *All of Me* excerpt, but with the octave displacements of pitch as shown in Figure 5.4. Octave equivalence is not employed within *Jam Factory's* transition tables and therefore, if both middle C and the C one octave above appear in an event sequence, the pitches are classified as two different events. Figure 5.23 shows the 1st-order transition table used in the second section, and a possible output in common notation.

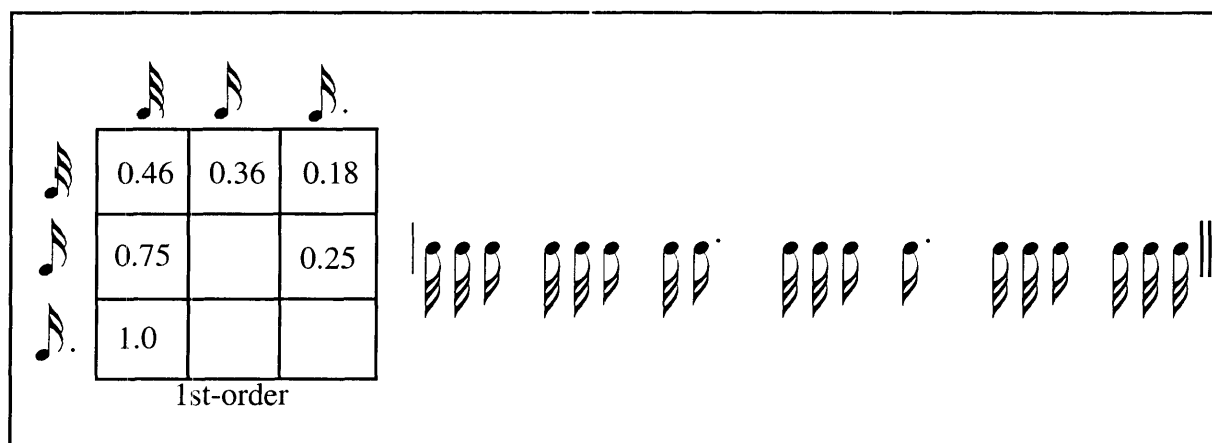
Figure 5.23 *Étude in Memoriam*, Part 1, section 2, pitch transition table and output



Rhythmic durations for the truncated bell tone in the second section of Part 1 are subject to a 1st-order transition table. As shown in Figure 5.6, the durations are from a newly-composed sequence in which there are three different values: a 1/32nd note, a 1/16th note and a dotted 1/16th note. Figure 5.24 shows the 1st-order transition table for the newly-composed sequence, and a possible rhythmic output. The numerous transitions toward the 1/32nd note value result in duration sequences with high proportions of 1/32nd note values,

consistently resembling the newly-composed duration sequence with its predominance of 1/32nd note values.

Figure 5.24 *Étude in Memoriam*, Part 1, section 2, duration transition table and output



Part 2 of the *Étude* employs 1st-order and 2nd-order transition tables with pitches from the full excerpt of *All of Me*, the transition tables previously shown in Figures 5.15 and 5.17. A combination of 50% 1st-order and 50% 2nd-order tables is used for the tongued trumpet sample, while 75% 1st-order and 25% 2nd-order combinations are used for the muted trumpet sample. The truncated bell tone in the section is assigned a 1st-order table, while the more melodic normal trumpet sample has a 2nd-order table that, while providing variations on the original excerpt, retains much of the original pitch ordering due to the generation of motifs such as those shown in Figure 5.18. Within Part 2 of the *Étude* the employment of 1st-order and 2nd-order transition tables, and the combinations of the tables, result in numerous variations on the original excerpt; however, the resulting pitch material is unified through the common use of the excerpt's single pitch sequence.

Part 3 of the *Étude* again uses the full pitch sequence from the excerpt, with the exception of the mono-tone used in the hammer and nail sample, and the three-note fragment used for the saw sample. Pitch transition tables are not employed with either of these timbres, the lack of repeated pitches precluding the use of transition tables. For the remaining timbres, a 1st-order transition table is assigned to the pitched hammer and nail

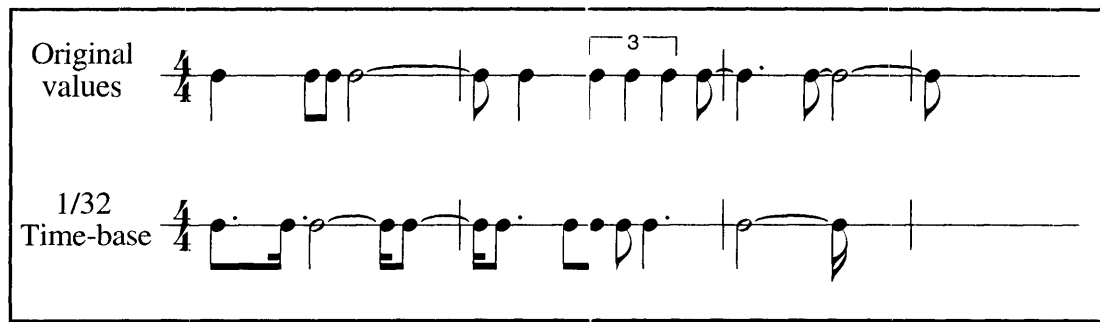
timbre, while a 2nd-order table is assigned to the truncated bell tone. These two timbres in combination present a *staccato* dialogue, their differing variations resulting from the different transition tables used. In the final section of Part 3, remaining timbres and their parameters are taken from previous sections in the work, and are discussed in a later section on the editing of the *Étude*.

Durations for the hammer and nail samples and the truncated bell tone in Part 3 are from the original sequence in the *All of Me* excerpt, and subject to variations with both 1st-order and 2nd-order transition tables. The pitched and mono-tone hammer and nail samples are assigned a 1st-order table, while the truncated bell tone is assigned a 2nd-order table. While the use of a single transition table in both the hammer and nail samples can produce different rhythmic variations between the two players assigned these timbres (each might follow a different probability path), different Time-base settings, as shown in Figure 5.8, and the use of Time-distortion, ensure rhythmic diversity. Figure 5.25 shows the 1st-order duration transition table. Figure 5.26 gives a possible output from the original *All of Me* duration series and the same output converted to 1/32nd note Time-base values, as used with the mono-tone nail sample.

Figure 5.25 *Étude in Memoriam*, Part 3, 1st-order duration transition table

					0.33	0.33		0.33
3		0.5	0.25		0.25			
	0.67	0.33						
	1.0							
					1.0			
		1.0						
			1.0					
				1.0				

Figure 5.26 *Étude in Memoriam*, Part 3, 1st-order output, original and 1/32 Time-base



5.3.3. The application of algorithms in real-time

As in *M*, many algorithm settings in *Jam Factory* are configurable in real-time, including for example, the various assignments of transition tables to players and Time-distortion algorithm settings. With the exception of Transposition, Volume and Play-enable settings, settings made in the design stage of each of the six *Étude* interactions remained unaltered throughout the operation stages.

An example of the alteration to Transposition algorithm settings occurs in the second section of Part 1, the third truncated bell tone gesture of the section rapidly moving through semitone transpositions, descending then ascending, to form a long gestural arc. Initial pitch material is the *All of Me* sequence with octave alterations as shown in Figure 5.4, subjected to a 1st-order transition table. Figure 5.27 illustrates the effect of rapid Transposition algorithm setting alterations, the possible 1st-order output shown in Figure 5.23 used as a basis for a series of descending transpositions. With each pitch, the range of transposition is extended by one semitone: the second pitch is one semitone lower than in the original sequence, the final pitch, as the 19th pitch, is 18 semitones lower than the original.

Figure 5.27 *Étude in Memoriam*, Transpositions Part 1, section 2, truncated bell tone



Jam Factory's Play-enable algorithm, like *M's* Play-enable algorithm, allows individual activation of *Jam Factory* players, providing real-time control over texture. In many instances, Play-enable is used in the *Étude* in combination with volume settings. As an example, in the opening section of Part 1 normal bell tones, apart from the repeated bell that opens the work, are inactive at the opening of the work, allowing a focus on the repeated bell tone. The remaining three players in the section are activated successively and, following the activation of each player, the player's volume settings are increased, in effect providing a fade in of each player. The end of the section reverses the process, the velocity settings for each player being decreased to provide a fade out before players are de-activated with Play-enable. A further example using the same process is applied to the truncated bell tone in the second section of Part 1. Three gestures are presented with this timbre, each gesture using the volume setting to fade the timbre in and out of the overall texture of the section.

5.4. Editing

Each of the six *Jam Factory* sections in the *Étude* was saved as an individual MIDI file and imported into the *Vision* sequencing program. *Vision* allowed the editing of the *Jam Factory* sections at various musical levels. At a deep level, the final form of the work was realised, the sequencer allowing the ordering of the six *Jam Factory* sections into a unified whole. In connecting the six sections, gestures from previous sections were copied and inserted into succeeding sections. As an example, the Part 2 tongued trumpet sample at the end of the first section originally did not continue into the second. To connect the two sections the sample was copied from the end of the first section and inserted into the start of the second, resulting in a continuous tongued trumpet timbre bridging the two sections. In other parts of the *Étude*, gestures were added with the sequencer in places where the original sections were 'lacking'. For example in the second section of Part 3, the original section, in being limited to *Jam Factory's* four voices, could not provide the textural density required for a climax in the work. Overlaid on the material of the original four voices is a copy of the material assigned to the sustained bell tone with slow attack and decay from Part 1. A simple

alteration of MIDI channels allowed a change of timbre while retaining the remaining parameter settings. In this final section of the *Étude*, the Part 1 material is assigned to the trumpet sample with *glissando* attack, previously used in Part 2. This trumpet sample provides an intense timbre, more suited to this climactic section of the work than the sustained bell tone from Part 1.

5.5. Summary

In the above examinations of *Jam Factory*'s algorithms, stochastically-based transition tables and Markov processes are seen to provide control over the musical parameters of pitch and rhythm within the program. An array of further algorithms provides deterministic and/or stochastic control over remaining musical parameters such as transposition, dynamics, and articulations. As in *M*, algorithms in *Jam Factory* provide a medium in which musical parameters can be controlled in real-time in the operation stage of the program, or established in the design stage and left to work without real-time manipulation. The wide range of variations in pitch and rhythm provided by the transition tables allows the establishment of transition table settings in the design stage of the *Étude*. In the operation stage the high degree of variations provided by the transition tables allows a focus on real-time manipulations of texture and dynamics within each interaction of the work.

As shown in the examinations of Markov processes, transition tables generate variations that are closely linked to original pitch and rhythmic materials. Within the *Étude in Memoriam Allan Dagg*, variations on the *All of Me* excerpt are stylistically linked to variations a jazz improviser might provide, a jazz improviser's variations generally having strong associations with original materials. In the *Étude in Memoriam Allan Dagg*, improvisation style variations from *Jam Factory*'s transition tables provide an ideal medium for a memorial to a jazz trumpeter.

CHAPTER 6 — *Symbolic Composer — Descendant Lines*

6.1. Overview

Descendant Lines (Track 3 on the accompanying audio CD) is again a highly programmatic work, a reflection on personal friendships developed between three Australians, each with markedly different genealogical backgrounds. It also portrays episodes in the lives of two of the Australians' migrant fathers, one from Croatia, the other from Argentina. The work is comprised of three main sections, each section programmatically portraying a common part of the two migrants' lives. The first section focuses on the disintegration of life in the migrants' countries of origin, the second focuses on the migrants' journeys, and the final section focuses on individual and familial growth in Australia.

The form of *Descendant Lines* is primarily determined by programmatic associations in the work, various associations within each of the work's three main sections forming numerous subsections as shown in Table 6.1. The table lists the programmatic associations throughout the work, while the measure numbers and rehearsal letters provided are in reference to the full score given in Appendix 2.

Table 6.1 *Descendant Lines*, Programmatic associations and form

SECTION 1		SECTION 2		SECTION 3	
mm.#		mm.#		mm.#	
1-15	The Croatian	54-63	Departure A	109-20	Argentinian growth D
16-18	Croatian lifestyle disintegration	64-74	Croatian journey B	121	Argentinian's son
19-27	Escape	75-9	Argentinian's partner	122-33	Croatian growth
28-9	The Argentinian	80-4	Croatian's partner	134	Croatian's son
30-53	Argentinian lifestyle disintegration	85-94	Argentinian's journey	135-65	New society E
		95-108	Arrival C		

The development of *Descendant Lines* was carried out in three compositional stages. The initial stage entailed the establishment of pitch and rhythmic materials. In the second stage, pitch and rhythmic materials were entered into the algorithmic program *Symbolic Composer*, and manipulated with numerous algorithms in each section of the work before compilation to MIDI files. The final stage involved the transferral of MIDI files to *Finale* notation software for editing, collating of sections, and output in a common notation score format.

The following sections relate the processes involved in each of the three compositional stages of the work. Pitch and rhythmic materials established in the initial stage are outlined in section 6.2. In section 6.3, algorithms are discussed according to their function within the *Symbolic Composer* program, as grammar-based algorithms or processing algorithms that include both deterministic and stochastic processes. Section 6.4 briefly outlines the editing of the notated score.

6.2. Composition Structures

6.2.1. Pitch structures

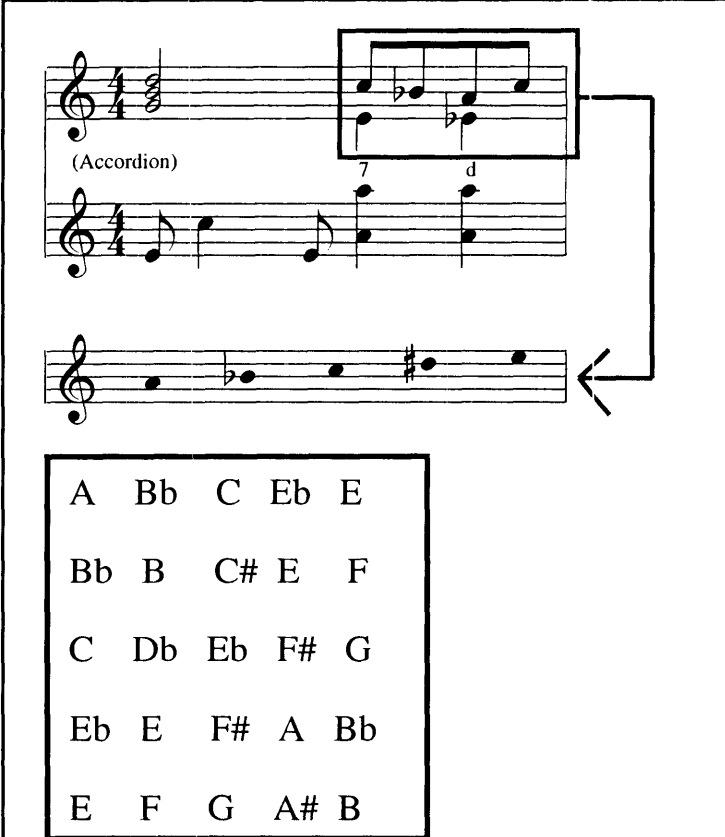
Pitch structures in *Descendant Lines* are derived from two sources, a traditional Croatian folksong *Tri Sam Leta Sa Tom Momice Seta* (Three Summers I've Walked With This Girl) and a Latin-American tango *Yo Te Quiero Mucho* (I Desire You Greatly). Fragments from the folksong and the tango are taken as starting points, the pitches within each fragment re-written in scale form as note sets. The note sets are subsequently placed in matrices similar to the note matrices used in the *Study for Triangles*, providing the transpositions used throughout the work. The following figures show the initial fragments, the note sets and matrices derived from the folksong (Figure 6.1) and the tango (Figure 6.2).

Figure 6.1 *Descendant Lines*, Pitch structure from *Tri Sam Leta Sa Tom Momice Seta*



B	C	D	E	F
C	Db	Eb	F	Gb
D	Eb	F	G	Ab
E	F	G	A	Bb
F	Gb	Ab	Bb	B

Figure 6.2 *Descendant Lines*, Pitch structure from *Yo Te Quiero Mucho*



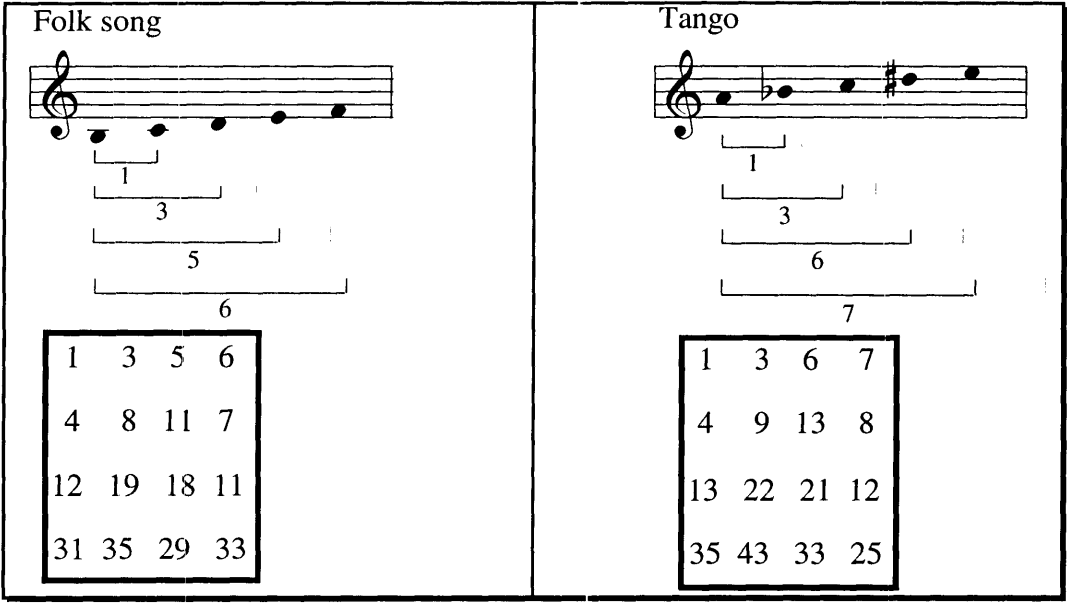
A	Bb	C	Eb	E
Bb	B	C#	E	F
C	Db	Eb	F#	G
Eb	E	F#	A	Bb
E	F	G	A#	B

Because of its being a work for piano accordions, *Descendant Lines* required further pitch structures to provide chordal parts for each accordion. These structures were based on the triadic sonorities produced by accordion buttons. The note sets derived from the folksong and tango form the basis of the work's left-hand chord structures, the note set derived from the folksong outlining a diminished triad of B-D-F, the note set derived from the tango outlining a minor triad of A-C-E. Throughout the work, left-hand chords produced by accordion buttons are limited to minor and diminished chords with root tones based on notes of the two note sets and their corresponding transpositions.

6.2.2. Rhythmic structures

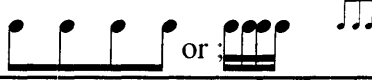
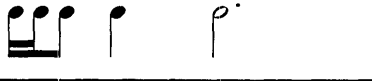
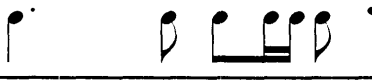
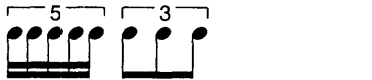
Two rhythmic structures used in *Descendant Lines* are derived from the intervallic content of the note sets shown in Figures 6.1 and 6.2. Interval distances (in semitones) are taken from the first note in each set. In the folksong set, the distance from the first note to the second is one semitone, to the third is three semitones, to the fourth is five semitones and to the last is six semitones, resulting in the numeric series 1 3 5 6. This series forms the first row of a matrix, successive rows derived with an additive process whereby the first two numerics of the initial row are added to form the first numeric of a second row. The second and third numerics are added to form the second numeric in the second row, the third and fourth are added to form the third numeric. The fourth and first numerics of the initial row form the final numeric in the second row. Figure 6.3 shows the numerics as derived from the two note sets and the derived matrices.

Figure 6.3 *Descendant Lines, Rhythmic structures*



In the tango matrix, each of the numeric units with values of 1 through to 9 are assigned a rhythmic cell in a list, these numerics generally indicating the number of rhythmic attacks within a cell, as shown in Table 6.2. Rhythmic cells are derived either from the tango, or are derived intuitively as newly-composed (original) cells. A similar assignment of rhythmic cells to numeric units occurs with the numerics derived from the folksong, each of the numerics 1-8 assigned a rhythmic cell.

Table 6.2 *Descendant Lines*, Numeric to rhythmic cell definitions

Numeric	Source*	Tango Cells	Numeric	Source*	Folksong Cells
1	T		1	FS	
3	O		3	FS	
4	T		4	O	
6	O		5	FS	
7	T		6	FS	
8	O		7	O	
9	O		8	O	

* T - Tango, FS - Folksong, O - Original

Numerics higher than 8 in the folksong matrix, and higher than 9 in the tango matrix, are combinations of various lower value numerics, and as such, represent different combinations of rhythmic cells. Within *Descendant Lines*, various combinations of higher value numerics are employed, for example the tango numeric 13 may appear as a combination of the numerics 8 + 5, 8 + 4 + 1 or 7 + 6, that is, any numeric combination whose sum is the higher numeric value. Lower value numerics are regularly represented with rhythmic augmentations or diminutions of the cells, examples of which are given in Table 6.2 in association with numerics 1 and 4 in both the tango and folksong rhythmic cells. In general, the assigning of numerics to various rhythmic cells allows a degree of rhythmic freedom with a limited set of rhythmic cells.

6.3. The Application of Algorithms

Some algorithms used in the development of *Descendant Lines* are deterministic, generating symbols according to various pre-defined mathematical or scientific processes.

Throughout the work, symbols generated by these algorithms are mapped to pitch and/or rhythm within *Symbolic Composer* ‘scores’. In the development of the work there were 20 separate scores compiled, one or two scores for each sub-section of the work. The scores used within the work are given in Appendix 3. Remaining musical parameters, while definable within *Symbolic Composer*, were manipulated in the final composition stage of the work in the notation software program *Finale*. Generative algorithms used in *Descendant Lines* fall into two main categories: algorithms based on grammars; and algorithms based on chaos theory.

6.3.1. Grammar-based algorithms

Grammars, as applied in computer science, arise from the study of linguistics, and in particular relate to syntax rules classified by Noam Chomsky in his *Syntactic Structures* (Chomsky 1957). Chomsky divides grammars into four main types, Type 0 to Type 3, each type utilising different rewriting rules. Within rewriting rules, also called productions, sets of ordered pairs of expressions are depicted as lhs $\rightarrow$ rhs: where symbols or items on the left side are rewritten ($\rightarrow$) as symbols/items on the right side. Type 2 grammars, also known as context-free grammars, are the most relevant to this study and are defined as a collection of three things:

1. An alphabet Σ of letters called **terminals** from which strings are made that will be the words of a language.
2. A set of symbols called **non-terminals**, one of which is the symbol *S*, standing for “start here”.
3. A finite set of productions of the form
one non-terminal $\rightarrow$ finite string of terminals and/or non-terminals

(Cohen 1991, p.251).

A simple linguistic example of this process is $S \rightarrow NP + VP$, where *S* is a sentence, *NP* is a noun phrase, and *VP* is a verb phrase. As described by Crystal (1971, p.227), a set of rewrite rules or productions could be:

<i>S</i>	$\rightarrow$	<i>NP + VP</i>
<i>VP</i>	$\rightarrow$	<i>V (Verb) + NP</i>
<i>NP</i>	$\rightarrow$	<i>T (Article) + N (Noun)</i>
<i>T</i>	$\rightarrow$	the
<i>N</i>	$\rightarrow$	boy, girl
<i>V</i>	$\rightarrow$	saw

The initial production states that a sentence is a combination of a noun phrase and a verb phrase. The second production states that the verb phrase consists of a verb and a noun phrase. The third production states that the noun phrase consists of an article and a noun. The output at this stage is that the sentence consists of T + N + V + T + N. As these symbols are further rewritten in the final three productions as concrete vocabulary items, they are called non-terminals. The remaining rules specify terminals, (i.e. the elements that cannot be rewritten), for example the article *the*, the nouns *boy* and *girl* and the verb *saw*. The final output produces the sentences *The boy saw the girl*, or *The girl saw the boy* (Crystal 1971, p.227).

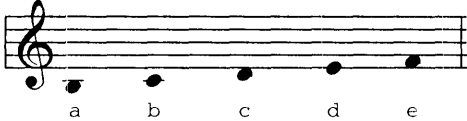
Increasing the number of vocabulary items in the last three rules increases the number of sentences the rules can generate. ‘Adding *man* and *kicked* can produce *The boy saw the man*, *The man kicked the girl*, and many others’ (Crystal 1971, pp.227-8). Further rules are added to introduce adjectives, auxiliary verbs, pronouns etc. to account for complexities in noun and verb phrases.

Two grammar-based algorithms were used in *Descendant Lines* and are named within *Symbolic Composer* as `gen-notrans` and `gen-rewrite`. Prior to implementations of these algorithms, productions, similar to $S \rightarrow NP + VP$, must be pre-defined within *Symbolic Composer*, the definitions made using a *Symbolic Composer* function called `defsymb`. The two note sets used in *Descendant Lines* map to symbols as shown in Figure 6.4(a). Using a rewriting rule, the `defsymb` function allows the expansion of each symbol as shown in the inner brackets of Figure 6.4(b). The symbols used for the expansion of initial symbols in each note set were subjectively chosen to produce note motifs within the work, the note motifs shown in square brackets in Figure 6.4(c).

Figure 6.4 *Descendant Lines*, Symbol to pitch mapping, symbol definitions and definition mappings

6.4(a)

Folksong symbols



Tango symbols



6.4(b)

Folksong Definitions

```
(defsym a '(e b d))
(defsym b '(f c))
(defsym c '(e a b))
(defsym d '(g b))
(defsym e '(d h f))
```

Tango Definitions

```
(defsym a '(b d))
(defsym b '(c d e))
(defsym c '(a g))
(defsym d '(e g))
(defsym e '(c b f))
```

6.4(c)

Folksong



Tango



The *Symbolic Composer* algorithm `gen-notrans` implements a rewriting process that incorporates both left-hand and right-hand production symbols given in the `defsym` definitions. When used in a *Symbolic Composer* score, the `gen-notrans` function is accompanied by a starting symbol, for example the symbol `a`, and a number that specifies the number of productions the algorithm will use. The definition `gen-notrans a 2` indicates that two production levels will occur, beginning with the symbol `a`. In comparison

to the former linguistics-based example, the symbol `a` in a `gen-notrans a 2` definition is similar to the symbol `S` standing for sentence, however the symbol `a` is also incorporated in the `gen-notrans` output. Each of the remaining symbols, for the purpose of this comparison, can be equated with alphabetical symbols as used in *Symbolic Composer*:

<code>S</code>	$\equiv$	<code>a</code>	(<code>S</code> is equivalent to <code>a</code>)
<code>NP</code>	$\equiv$	<code>b</code>	
<code>VP</code>	$\equiv$	<code>c</code>	
<code>V</code>	$\equiv$	<code>d</code>	
<code>T</code>	$\equiv$	<code>e</code>	
<code>N</code>	$\equiv$	<code>f</code>	

The production rules, as seen in the linguistics example, translate to:

```

a → b c (a is rewritten as b c)
c → d b
b → e f

```

This translation, in *Symbolic Composer's* LISP language, is written as:

```

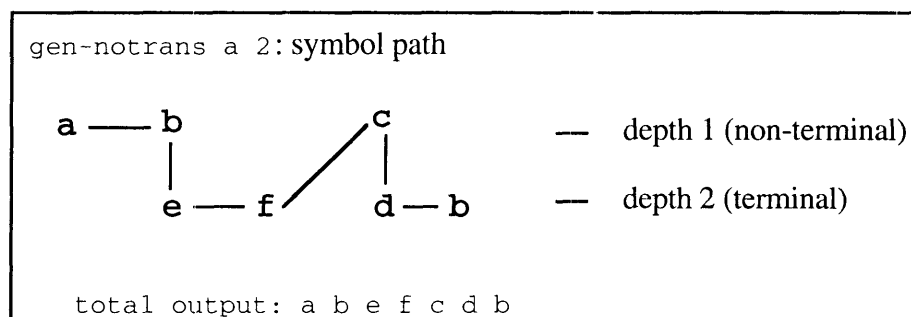
(defsym a '(b c))
(defsym c '(d b))
(defsym b '(e f))

```

The most simple `gen-notrans` function (`gen-notrans a 1`) takes into account the symbol definitions defined with `defsym`, outputting with the above definitions the symbol sequence `a b c`. When the function is specified as `gen-notrans a 2`, the remaining production rules are implemented on two depth levels, the output incorporated into the `gen-notrans a 1` output, as shown in Figure 6.5a. Like the linguistics example that specifies terminals as particular words, `gen-notrans` allows terminals with the symbols that are not rewritten in the rewrite rules. In the above example the terminals are the symbols `d`, `e` and `f`. At the point in `gen-notrans` where these terminals are reached, the algorithm moves onto the next non-terminal symbol, continuing the generative process. In moving onto the next symbol after a terminal, `gen-notrans`, unlike the linguistics example, can provide indefinite applications of rewrite rules. In *Symbolic Composer*, indefinite rewriting occurs until a terminating condition is invoked, the numeric `2` specifying the terminating condition

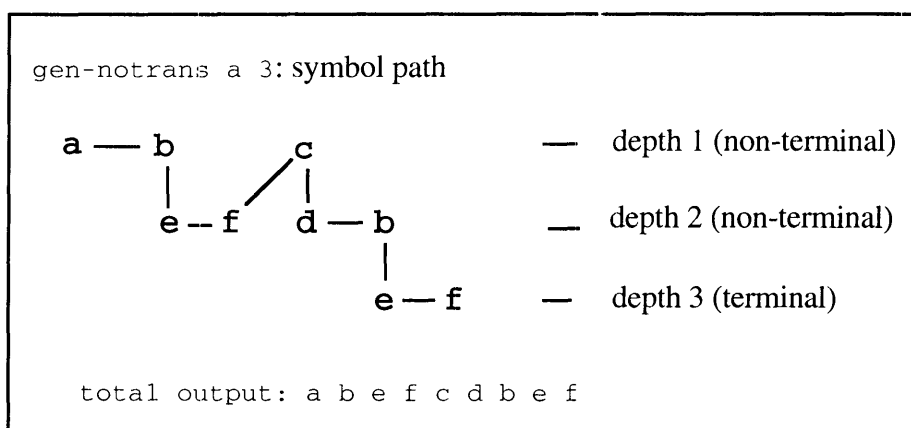
that symbols generated at the second depth level terminate the algorithm. In the following figures, lines between symbols indicate chronological movement of symbol events.

Figure 6.5a *Symbolic Composer*, Symbol output, `gen-notrans a 2`



Symbolic Composer allows unlimited depths for the `gen-notrans` algorithm. Figure 6.5b shows the symbol path for `gen-notrans a 3`, again using the previous `defsymb` definitions.

Figure 6.5b *Symbolic Composer*, Symbol output, `gen-notrans a 3`



Within *Descendant Lines* the `gen-notrans` algorithm generates symbols for pitch material for the majority of the work. In the opening section, the right-hand parts of both accordions are assigned various `gen-notrans` depths and starting symbols, generating symbols from the folksong symbol definitions as shown in Figure 6.4 (b). The `gen-notrans` algorithm is implemented within the `define instrument symbol` chunk as shown in Figure 6.6, the Figure also showing the symbol paths for each `gen-notrans` setting.

Composer alphanumeric nomenclature, using lower case letters in the Courier font, is used throughout this chapter to denote specific pitch and tessitura, whilst upper case letters apply to notes).

```
create-tonality accord.set2 '(b 4 c 5 d 5 e 5 f 5))
```

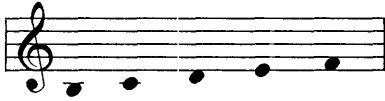
The `activate-tonality` function allows the specification of note sets used in the opening section, and the starting pitches of the sets:

```
(activate-tonality (accord.set2 f 6) (accord.set2 e 6))
```

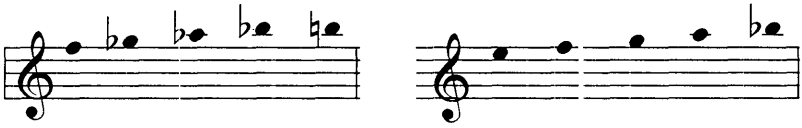
In the opening sub-section, two transpositions are used. The final or fifth transposition from the folksong note matrix (see Figure 6.1) is the first, starting on the pitch `f 6`, which is followed by the fourth transposition starting on the pitch `e 6` in measure 9 of the work. Change points for note sets are specified with the word `changes` in *Symbolic Composer's* timesheet, a full stop indicating the change point in relation to the timesheet's measure indicator. Figure 6.7 shows the notation of the folksong note set as defined with the `create-tonality` function, the transpositions as set with the `activate-tonality` function, and the change point as used in the timesheet for the opening section. The vertical and horizontal lines in the timesheet are a graphic indication of measures and beats, the vertical lines representing the first beat in a measure, the horizontal lines indicating the remaining beats.

Figure 6.7 *Descendant Lines*, Opening section note definitions

```
(create-tonality accord.set2 '(b 4 c 5 d 5 e 5 f 5))
```



```
(activate tonality (accord.set2 f 6) (accord set2 e 6))
```



Timesheet

```

;measures      |---|---|---|---|---|---|---|---|---|---|---|---|---|---|
changes tonals1 "

```


with `defsym`. An examination of the three `gen-notrans` symbol sequences in Figure 6.8 reveals a consistency of symbolic motifs appearing in all sequences. For example, the motif `b f c` appears once in the `gen-notrans e 4` sequence, once in the `gen-notrans a 2` sequence and four times in the `gen-notrans a 4` sequence. When mapped to the folksong note set, this motif provides an interval set of 11 semitones from `b` to `f`, and six semitones from `f` to `c`. In every transposition of the note set this interval set remains, providing intervallic consistencies within pitch materials based on the folksong note set. Programmatically, the consistencies provided by the sequences generated with `gen-notrans` are collectively viewed as a musical identity for the Croatian migrant. Similarly, tango definitions are implemented with `gen-notrans` functions, the resulting sequences containing motifs that are collectively viewed as a musical identity for the Argentinian migrant.

The `gen-notrans` algorithm is also applied to rhythm within *Descendant Lines*, the algorithm generating sequences of the rhythmic cells used in the work. Within *Symbolic Composer*, the *Descendant Lines* rhythmic cells are defined as having a relationship with single symbols. Durations are defined in the program with fraction representations of American time names ($1/4$, $1/8$, $1/16$ etc). Tuplets are similarly defined but with an extension of the normal fraction representation, $1/4-5$ for example, representing a $1/16$ th note quintuplet, through the division of the $1/4$ note value into five durations.

Figure 6.9 shows the accordion 1 rhythmic definitions for the first sub-section in the second section of the work (mm.54-63). The function `def-neuron` allows the establishment of rules based on if/then operators. The name `sym-to-len` describes the process of converting a symbol to a length pattern, or in the case of *Descendant Lines*, a rhythmic cell. The rhythmic cells defined in Figure 6.9 relate to the numeric series 1 3 6 7 and 4 9 13 8, the rhythmic cells and numerics derived from the tango. All but the last numeric (8) are defined for this section of the work. Following the `sym-to-len` definitions in Figure 6.9 is a definition for a sequence of rhythmic cells produced by the symbols: the word `rhy1`. The function `run-neuron` implements the symbol to length conversion, a symbol sequence then provided by `gen-notrans a 3`. In this section, `gen-notrans a 3` uses the `defsym` definitions for the tango shown in Figure 6.4 (b). The output of `gen-notrans a 3` is shown

in Figure 6.9, along with some of the resulting rhythms in common notation. The overall output, defined as `rhy1` is assigned to accordion 1 in the chunk for instrument rhythm definitions, `def-instrument-length`.

Figure 6.9 *Descendant Lines*, mm.54-63, `gen-notrans` with tango rhythmic cells

```
(def-neuron sym-to-len
(in 1 'a) '(1/2)
(in 1 'b) '(1/4-3 1/4-3 1/4-3)
(in 1 'c) '(-1/4-6 1/4-6 1/4-6 1/4-6 1/4-6 1/4-6)
(in 1 'd) '(1/16 1/8 1/16 1/16 1/16 1/16 1/16)
(in 1 'e) '(1/16 1/16 1/16 1/16)
(in 1 'f) '(1/4-5 1/4-5 1/4-5 1/4-5 1/4-5 1/4-3 1/4-3 1/4-3 1/4)
(in 1 'g) '(1/4-7 1/4-7 1/4-7 1/4-7 1/4-7 1/4-7 1/4-7 1/4 1/16 1/16 1/16 1/16 1/4)
)
(setq rhy1 (run-neuron 'sym-to-len (gen-notrans a 3)))
)
(def-instrument-length
accord1 rhy1
)
```

output: gen-notrans a 3

a b c a g d e g e c b f e c b f g

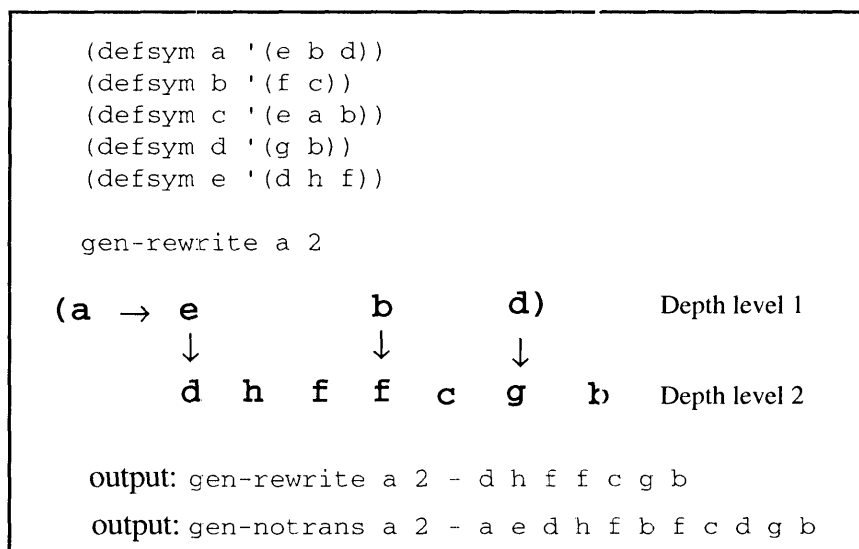
The second grammar-based algorithm used in *Descendant Lines* is `gen-rewrite`. This algorithm implements strict rewriting rules in which the left-hand side symbol of a production is always replaced by right-hand side symbols, in the same way as the `S` in the former linguistics-based example is replaced by `NP + VP`. Like `gen-notrans`, `gen-rewrite` represents a Type 2 grammar in Chomsky's classifications, a context-free grammar that has productions with a single left-hand side symbol and finite right-hand side symbols.

The generative process made available with `gen-rewrite` is closely related to Lindenmayer systems, or L-systems, which originated in the work of Aristid Lindenmayer. Lindenmayer 'introduced language-theoretic models for developmental biology in 1968' (Rozenburg & Salomaa 1985, p.v), and the models have been applied in a range of fields such as biology, mathematics and computer science. Unlike grammars, L-systems do not

distinguish between terminals and non-terminals: ‘repeated generation by the application of recursive rewrite rules may continue indefinitely until some terminating condition is reached.’ (Jones 1989, p.187). In this respect, `gen-notrans` may also be considered as a close relation to an L-system in that the production rules defined with `defsym` and implemented with `gen-notrans` can indefinitely generate symbols until the terminating condition (a specified depth level) is reached.

Like `gen-notrans`, `gen-rewrite` also implements rewriting rules defined with `defsym`, specifications for the algorithm similarly allowing the definition of a starting symbol and a terminating condition or depth level. Figure 6.10 shows a `gen-rewrite` symbol output generated with the `defsym` folksong definitions, a starting symbol `a` and a depth level of 2. The Figure also provides the output of `gen-notrans a 2` for comparison. The difference between the two algorithms is that where `gen-notrans` *expands* the left-hand side symbols listed in `defsym` with right-hand side symbols, `gen-rewrite` *replaces* the left-hand side symbols with right-hand side symbols. Arrows in Figure 6.10 indicate the symbol replacements occurring with `gen-rewrite`.

Figure 6.10 *Symbolic Composer*, `gen-rewrite`, with folksong symbols



Within *Symbolic Composer*, the `gen-rewrite` algorithm is described as a method ‘developed to model the growth of organic systems’ (Thomas & Morgan 1990, p.71). As a

model for an organic growth system, the algorithm was used in *Descendant Lines* to generate pitch material as a programmatic identity for the Croatian migrant's son and uses the same `defsym` rewrite rules used with `gen-notrans` that produced a pitch identity for the Croatian migrant. The `gen-rewrite` algorithm is initially used in the accordion 2 solo at m.134 of the work, this sub-section programmatically depicting the migrant's son. As shown in Figure 6.11, three separate symbol sequences are derived from three depth levels of `gen-rewrite`: a 2, a 4 and a 6. A fourth symbol sequence is comprised of the symbols a b c d e, and is derived from the *Symbolic Composer* score for mm.80-84 which is used to represent the Croatian migrant's partner. When mapped to pitch, the symbol sequences provide a combination of programmatically-based pitch identities from the migrant and his partner, the migrant's identity transformed with `gen-rewrite` to form a new pitch identity for his son. The `gen-rewrite` pitch material is presented in m.134 in the accordionist's right-hand with the third transposition from the folksong note matrix set starting on C, while the symbol sequence a b c d e is presented in the left-hand, this sequence starting on B.

Figure 6.11 *Descendant Lines*, m. 134, `gen-rewrite` outputs and implementation

```

gen-rewrite a 2: output; d h f f c g b
gen-rewrite a 4: output; f c d h f e b d f c e a b
gen-rewrite a 6: output; d h f e b d f c f c g b e a b f c d h f
                        e b d f c g b d h f f c g b e a b

```

gen-rewrite a 6

* Final editing of the lh part includes octave transpositions and pitch alterations

Within the m.134 sub-section, `gen-rewrite` symbols are also used to generate rhythmic cells, in exactly the same way that `gen-notrans` symbols were previously seen to be converted to rhythmic cells. In this sub-section the numeric series 1 3 5 6 and 4 8 11 7 are used, each numeric assigned a folksong-based rhythmic cell. The rhythmic cells are then mapped to the symbols a to h, one symbol for each successive numeric in the two series (a = 1, b = 3, c = 5 etc.) The output from `gen-rewrite a 6` provides the order of the rhythmic cells throughout the sub-section. As an example, the initial two symbols generated by `gen-rewrite` are d and h. The symbol d is converted in *Symbolic Composer* to the rhythmic cell for the numeric 6 (1/4 . 1/8 1/8 1/16 1/16 1/8), and the symbol h is converted to the septuplet rhythmic cell for the numeric 7 (1/4-7 1/4-7 1/4-7 1/4-7 1/4-7 1/4-7 1/4-7). The two rhythmic cells in sequence provide the primary rhythmic values that begin both the left and right-hand parts of the m.134 sub-section, as shown in Figure 6.11.

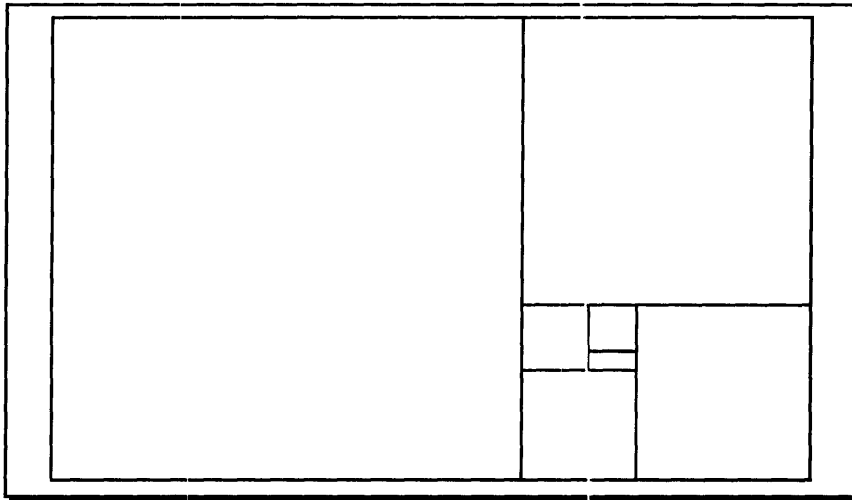
A third generative algorithm used in *Descendant Lines* is `gen-fibonacci`, an algorithm implementing the Fibonacci number series named after Leonardo Fibonacci of Pisa (c. 1170-1250), ‘a number theorist and algebraist, who first introduced the Arabic number system to Europe’ (Borowski & Borwein 1989, p.220). The Fibonacci number series is the sequence of integers 0 1 1 2 3 5 8 13 21 34..., the integers in the series moving toward infinity.

The ratio of successive integers tends to the *golden ratio* wherein

$$G = \frac{\sqrt{5} - 1}{2} = 0.618033988$$

It is a consequence of the definition that if one draws a rectangle with sides in the golden ratio (a *golden rectangle*), and then removes from it a square, the rectangle that remains has the same proportions as the original. If this process is repeated [see Figure 6.12] then the successive points of division lie on a logarithmic spiral (Borowski & Borwein 1989, p.249).

Figure 6.12 Golden ratio, sequence of golden rectangles



The Fibonacci series has been widely applied, most recently in computer science, and historically appears, for example, in the work of Kepler in 1611 ‘in connection with “phyllotaxis”, the study of the arrangement of leaves and flowers in plant life’; in the work of the mathematicians G. Lamé in 1844, and E. Lucas in 1878 (Knuth 1973, p.79); and more recently in the work of the architect Le Corbusier (Matossian 1986, p.40). ‘In the art world the ratio is said to be the most pleasing proportion aesthetically’ (Knuth 1973, p.79). Numerous composers have applied the Fibonacci series in their works, notably Debussy (Parks 1989, pp.209-10), Xenakis (Matossian 1986, p.60), Stockhausen and Nono (Smith-Brindle 1987, p.48).

The `gen-fibonacci` algorithm in *Symbolic Composer* outputs a symbol pattern based on two symbolic seed patterns and a depth integer. While this algorithm is not grammar-based, it can incorporate `gen-notations` patterns (or any other patterns) as seed patterns, and as such is included in this section of the study. Figure 6.13 provides an example of the algorithm. The single symbols `a` and `b` form two seed patterns, and are assigned a depth pattern which corresponds to a Fibonacci series integer.

Figure 6.13 *Symbolic Composer*, gen-fibonacci settings and outputs

(gen-fibonacci 1 'a 'b) output: ab	(Fibonacci series integer 1)
(gen-fibonacci 2 'a 'b) output: aba	(Fibonacci series integer 2)
(gen-fibonacci 3 'a 'b) output: abaab	(Fibonacci series integer 3)
(gen-fibonacci 4 'a 'b) output: abaababa	(Fibonacci series integer 5)
(gen-fibonacci 5 'a 'b) output: abaababaabaab	(Fibonacci series integer 8)

Within *Descendant Lines* the two seed symbol patterns for gen-fibonacci are either gen-notrans patterns, or symbolic chord patterns for the accordion left-hand parts. Figure 6.14 shows the output of gen-fibonacci using gen-notrans a 4 and a 3 based on the defsym tango symbols, and the output of a symbolic chord pattern in which a single bass note is followed by a chord in two separate seed patterns. The gen-fibonacci settings are applied in m.121 of *Descendant Lines*, the gen-notrans symbols applied in the right-hand part and mapped to the tango note set without transposition, while the left-hand chord symbols are mapped to a C chromatic scale.

Figure 6.14 *Descendant Lines*, m.121, gen-fibonacci outputs, mappings and implementation

```
(gen-fibonacci 2 (gen-notrans a 4) (gen-notrans a 3)))
```

output:

```
a b c a b d g d e c b f g e c a g
b c d e f d e c a g b c d e f g      = (gen-notrans a 4)
```



```
a b c a g d e g e c b f d e c b f g = (gen-notrans a 3)
```



```
a b c a b d g d e c b f g e c a g
b c d e f d e c a g b c d e f g      = (gen-notrans a 4)
```



```
(setq chords (gen-fibonacci 6 '(a dgkp) '(j adhm)))
```

output:

```
a dgkp j adhm a dgkp a dgkp j adhm a dgkp j adhm a dgkp a dgkp
j adhm a dgkp a dgkp j adhm a dgkp j adhm a dgkp a dgkp j adhm
a dgkp j adhm a dgkp
```


Mappings:

```
tango pitch set - A Bb C Eb E A Bb C
                  a b c d e f g h
```



```
C chromatic scale - C Eb Gb Bb Eb (C bass note + Eb minor)
                   a d g k p (pattern a)
```



```
A C Eb G C      (A bass note + C minor)
j a d h m        (pattern b)
```


_____ gen-notrans a 4 _____

_____ gen-notrans a 3 _____ gen-notrans a 4 _____

With the depth setting of 2, the *gen-fibonacci* algorithm as applied in the right-hand part of m.121 is, in effect, only a statement of the two *gen-notrans* patterns used, and

not particularly representative of the algorithm's capability. The material in the left-hand part is more representative with its depth level of 6, even though the full output at this depth is not presented in the part. Here the final three symbols are not included due to the length of the sub-section, and its rhythmic definitions as described below.

Notable in the left-hand part of m.121 are the incomplete statements of the chord symbol patterns: each pattern marked a or b in Figure 6.14 should include a bass note followed by a minor chord. The variation is due to the use of a different set of rhythmic cells assigned to the numeric series derived from the tango note set. In m.121, the numeric series used are 1 3 6 7 and 4 9 13. Within the rhythmic cells, each numeric of the series is defined in 1/16th note values, the first 1/16th stated as a note, the remaining 1/16ths as rests:

1 = , 3 = , 6 = etc.

As in the m.134 sub-section, the cells in m.121 are converted to symbols. Symbol patterns for the cells are then generated with `gen-notrans a 2` using the `defsym` tango definitions, as shown in Figure 6.15. Wherever defined bass notes or chords coincide with a rest, the rest appears in the score.

Further rhythmic spacing in the part is controlled with *Symbolic Composer's* timesheet. The timesheet for m.121 has a resolution of 1/4. The timesheet dashes, as shown in Figure 6.15, appear immediately below the timesheet's graphic measures indicator. The dashes indicate where the rhythmic cells are active for the duration of a 1/4 note, while the absence of a dash indicates a 1/4 note rest.

Figure 6.15 *Descendant Lines*, m.121, left-hand part rhythmic definitions, output and timesheet

```

(def-neuron sym-to-len2
  (in 1 'a) '(1/16)
  (in 1 'b) '(1/16 -2/16)
  (in 1 'c) '(1/16 -5/16)
  (in 1 'd) '(1/16 -6/16)
  (in 1 'e) '(1/16 -3/16)
  (in 1 'f) '(1/16 -8/16)
  (in 1 'g) '(1/16 -12/16)
)

(run-neuron 'sym-to-len2
  (gen-notrans a 2))

output: a b c d e d e g

```



```

Timesheet:
;MEASURES          |---|---|---|---|---|---|---|---|---|---|
accord1lh tonals2  "--  ---  ---  ----  -  -----  ----  "
)

```

The `gen-fibonacci` algorithm used in the m.121 sub-section functions to provide a programmatic identity for the Argentinian's son. The left-hand part is based on materials presented earlier in the work, in mm.28-29, materials that previously provided an identity for the Argentinian. With the new rhythmic cells, and pitch and rhythmic materials generated with `gen-fibonacci`, the left-hand part provides a new context for the right-hand part, a context that provides a programmatic identity for the Argentinian's son.

6.3.2. Chaos-based algorithms

Chaos theory reveals the way in which 'many systems that are constantly changing are extremely sensitive to their initial state -- position, velocity, and so on. As a system evolves in time, minute changes amplify rapidly' (Hall 1991, p.8). An example that shows both non-chaotic and chaotic systems involves the motion of a spherical pendulum — one that is free to move in any direction:

If the system is driven with a smooth, periodic, horizontal motion at the pivot, the bob will start to swing about. After a while, it may settle down into a stable and entirely predictable pattern of motion (a non-chaotic system), in which the bob traces out a roughly elliptical path with the same period as the driving force (Davies & Gribbin 1991, pp.28-9).

If the position and velocity of the bob at some point are measured, subsequent behaviour can be predictively calculated. 'If the driving frequency in this system is altered slightly, the regular motion of the pendulum may (if the frequency is close to some critical value) give way to chaos, with the bob swinging first this way and then that' (Davies & Gribbin 1991, p.29). If the position and velocity of the bob at some point are again measured, predictive calculations concerning subsequent behaviour will contain errors that grow exponentially with time. This is the hallmark of a chaotic system.

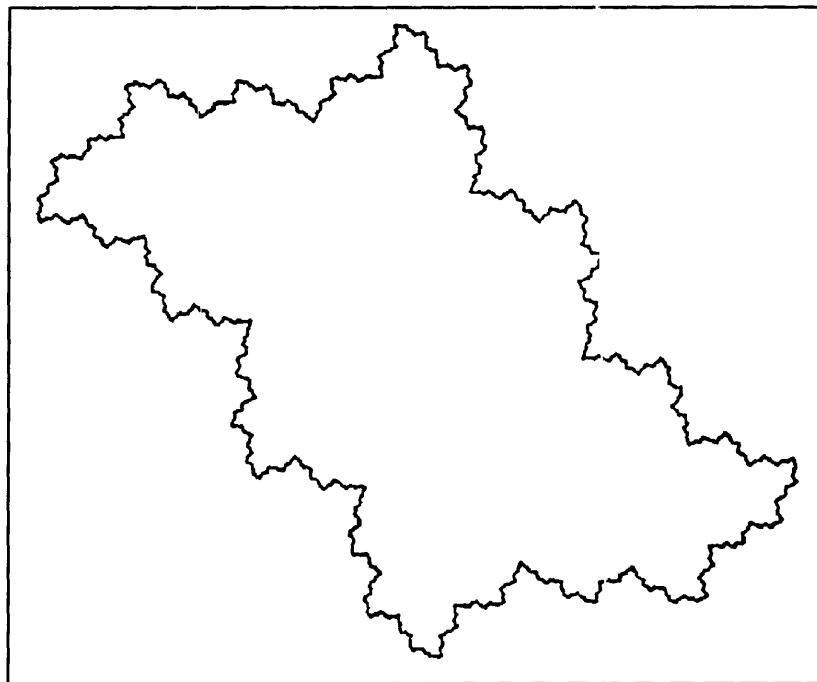
Chaos theory has developed with applications in a wide range of scientific fields, providing 'an interpretive tool for describing many of the complexities of the world, from brain rhythms to gold futures' (Hall 1991, p.9). It is used in astronomy to model the pulsation of the early universe and the motion of stars in galaxies, in the study of the aurorae, in the movements of the atmosphere and weather, in biology, physics, engineering and so on (Percival 1991, p.16). The computer has revealed the subtle behaviour of chaotic systems because it can follow their trajectories over many millions of steps. Computer graphics associated with chaotic systems show how quite simple mathematical equations can produce fractals, patterns of ever increasing complexity (Hall 1991, p.9).

Within *Descendant Lines*, three of *Symbolic Composer's* chaos-based algorithms are employed: `gen-hopalong-symbol`, `gen-noise-brownian` and `gen-lorenz`. As the prefixes imply, these algorithms are generative; however, unlike grammar-based algorithms that generate symbols, they are numerically based and generate vector patterns containing integers. The first two algorithms, `gen-hopalong-symbol`, and `gen-noise-brownian`, relate to chaos-based fractal images. The third algorithm, `gen-lorenz`, relates to a chaotic system originally used for modelling weather patterns.

The term fractal was coined by the French mathematician Benoit Mandelbrot in 1975 to describe computer generated, self-similar graphic images produced by a mathematical

process using iterations of polynomials. An expression of this process is $z \rightarrow z^2 + c$ where a value z evolves over a series of steps (indicated by the arrow) so that the new value of z is the square of the previous of z plus a constant c . In fractal geometry, z and c may be complex numbers, the complex number variables of x and y defining a plane in which z moves with successive iterations. The movement of z in this plane is similar to the chaotic movement of the pendulum described above. 'Computer modelling can be used to follow the movement of z .' (Series 1991, p.137). As an example, the constant c , where $c = -0.12375 + 0.56508i$, will produce a Julia set (1918), named after the French mathematician Gaston Julia. The resulting computer graphic, as shown in Figure 6.16, relates self-similarity: 'If we look at any one of the corners or bays, we notice that the same shape is found at another place in another size' (Peitgen & Richter 1986, p.10).

Figure 6.16 Julia Set (Pietgen & Richter 1986, p.10)

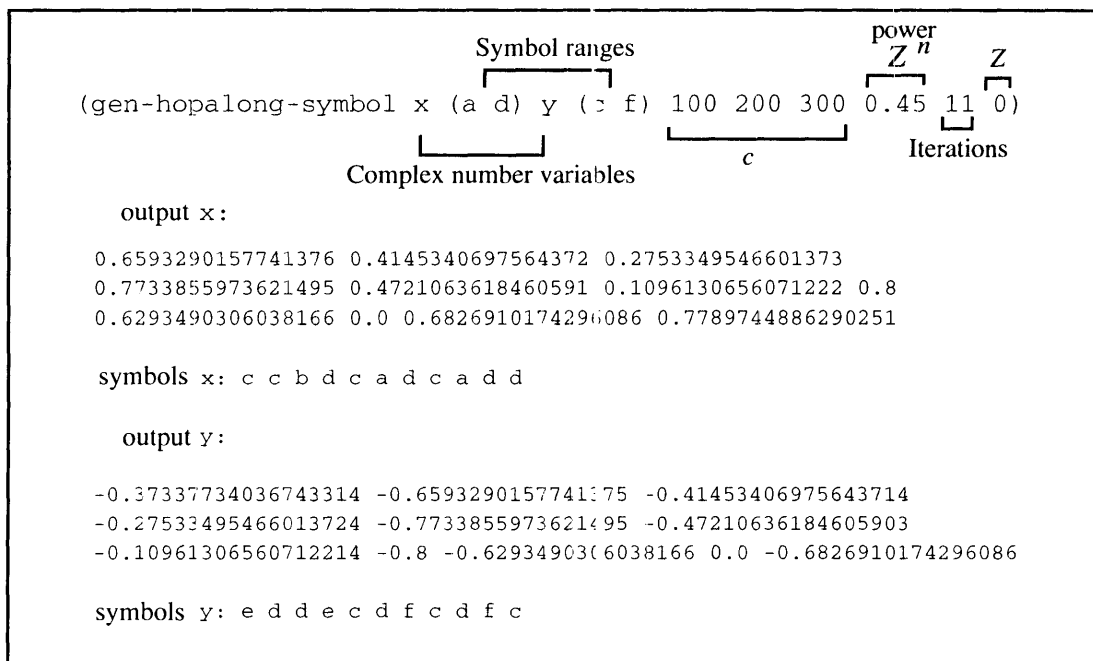


With the aid of computers, Mandelbrot realised the work of Julia in graphic form and proceeded to develop the famed Mandelbrot set. The primary difference between this set and the Julia set is that 'Julia sets are defined by fixing c and letting z vary in the field of

complex numbers, while the Mandelbrot set is obtained by fixing z and varying the parameter c ' (Douady 1986, p.161).

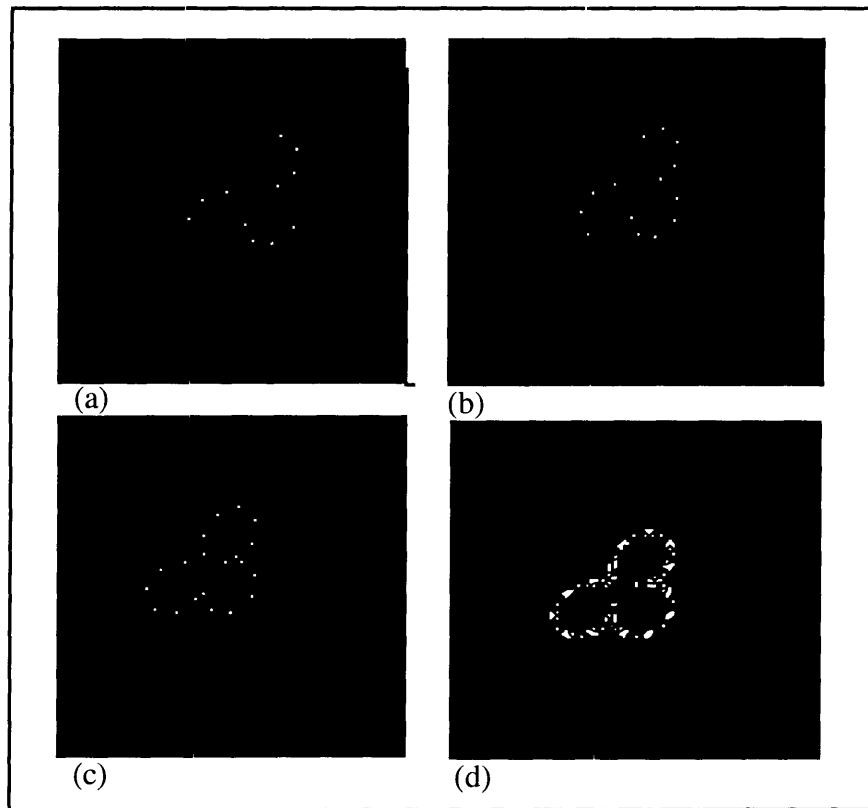
The `gen-hopalong` algorithm in *Symbolic Composer* allows the setting of z , a power, three constants, and the number of iterations the algorithm will use. The algorithm is related to the Mandelbrot set in that z is fixed, while the three constants allow variations on the parameter c . As a vector generator, the algorithm outputs numerics to x and y co-ordinates in the field of complex numbers. The numerics are mapped to symbols according to user defined symbol ranges for x and y , as shown in Figure 6.17.

Figure 6.17 *Symbolic Composer*, `gen-hopalong-symbol` algorithm, settings and output



Symbolic Composer's 'Visualiser' allows numerical output of program data to be viewed graphically. The graphic output of the Figure 6.17 `gen-hopalong` setting is shown in Figure 6.18(a), the 11 iterations resulting in a simple graphic. Higher values in the iteration setting result in more representative fractal images as shown in Figure 6.18(b) with 14 iterations, 6.18(c) with 20 iterations, and 6.18(d) with 200 iterations. Literal self-similarity, as shown by the different iterations of the algorithm, is visually evident.

Figure 6.18 *Symbolic Composer*, gen-hopalong algorithm, Visualizer representations



In *Descendant Lines*, the `gen-hopalong-symbol` algorithm is employed in sub-sections 2 and 3 of the opening section (mm.16-27). The settings for the algorithm are those shown in Figure 6.17. In the second sub-section (m.16) the symbolic output of the x variable is assigned to accordion 1, while the output of the y variable is assigned to accordion 2. The symbols are mapped to the folksong note set starting on D, the 11 iterations of the algorithm resulting in 11-note pitch sequences from the x and y variables. A 12th pitch in each accordion part is a repetition of the starting pitch.

Programmatically, the material in m.16 is associated with the disintegration of the Croatian's lifestyle. The programmatic identity of the Croatian, established in mm.1-15 with the folksong note set material and `gen-notrans`, is presented within a texturally calm context and in a slow tempo. In m.16, various parameter settings serve to present a more chaotic portrayal of the former material: the pitch material results from the 11 iterations of the chaos-based `gen-hopalong-symbol` algorithm; differing folksong rhythmic cells assigned to the two accordions result in rhythmic complexities with 4 against 5, 5 against 3

and 3 against 2; there is a dynamic shift from the former *ppp/pp* to *f*; and tempo is more than doubled.

The second part of the mm.16-18 sub-section (mm.17-18) condenses the chordal movement of mm.1-15, the folksong septuplet rhythmic cell used to present patterns of three diminished chords. A free *accelerando* in both parts serves to provide a continuation of the chaotic movement of the former measure.

The following sub-section (mm.19-27) employs a *Symbolic Composer* processing algorithm `symbol-transform`. Within this sub-section this algorithm transforms symbols produced by `gen-hopalong` into symbols produced by `gen-notrans`. The accordion 1 part again uses the *x* variable output from the `gen-hopalong` algorithm with the same settings as the previous sub-section, while the accordion 2 part uses the *y* variable. The `symbol-transform` algorithm is used to gradually transform the `gen-hopalong` symbols to `gen-notrans` symbols that are based on the `defsymb` folksong definitions. Figure 6.19 shows the symbol output of `gen-notrans` and `gen-hopalong`, and the result of applying the `symbol-transform` algorithm in the accordion 1 part. The `gen-notrans` symbols are shown in bold type, the remaining symbols are from `gen-hopalong`.

Figure 6.19 *Descendant Lines*, Accordion 1, mm.19-27, `symbol-transform` resulting output

```
gen-notrans e 4: e d g b f c e a b h f
gen-hopalong-symbol (x): c c b d c a d c a d d
symbol-transform
e c b d c a d c a d d e d g d c a d c a d d e d g b c a d c a d d e
d g b f c d c a d d e d g b f c e c a d d e d g b f c e a b d d e d g
b f c e a b h d e d g b f c e a b h f
```

The mm.19-27 sub-section programmatically represents the migrant's escape from Croatia. The chaos represented with the `gen-hopalong` material is dissipated as the escape is made, the success of the escape portrayed with the return of the Croatian's programmatic identity material formed in mm.1-15 with `gen-notrans`.

The second fractal-based algorithm used in *Descendant Lines* is `gen-noise-brownian`. Brownian noise is characteristic of Brownian motion, random movements of small particles suspended in a liquid and buffeted by the thermal agitation of molecules, first observed in 1827 by the botanist Robert Brown:

The motion of a 'Brownian particle' as seen under the microscope consists apparently of steps in a random direction and with a step-length that has some characteristic value, and therefore *random walk* is a term often used in the context of Brownian motion (Feder 1988, pp.163-4).

In general, a random walk consists of steps of adjacent values, analogous to:

the image of a drunk person who falls down, stands up, takes one step in a random direction, falls down again, stands up, takes another step in a random direction and so on (Moore 1990, p.442).

In relation to fractals, Mandelbrot has shown that nature provides many examples of self-similarity (Mandelbrot 1991, p.124). However, unlike the literal self-similarity seen in relation to the `gen-hopalong` algorithm, self-similarity in nature tends toward statistical self-similarity wherein the overall characteristics of a formation at a large scale are reflected in similar characteristics at a small scale (Jones 1989, p.180). As an example, a coastline or mountain range profile is self-similar when viewed from a distance, or from a few feet away. Such shapes as they appear in nature, along with the shapes of ferns, tree branches, rivers, lightning etc. approximate random walks, and are modelled with random fractals.

Symbolic Composer's `gen-noise-brownian` algorithm has three user definable numeric settings: `gen-noise-brownian 5 0.3 0.5`. The first allows the setting of a level that controls the number of output values the algorithm will generate: level 1 produces three values (ratio of 2-1); level 2 produces five (4-1); level 3 produces nine (8-1); level 4 produces 17 (16-1); level 5 produces 33 (32-1) and so on. The second setting initialises the random number generator, simply providing a random starting value for the algorithm. The third setting provides control over the random walk implemented by the algorithm, high settings resulting in the normal adjacent step-wise motion of the random walk, while low

settings introduce skips into the random walk, with the lowest setting resulting in continual skips.

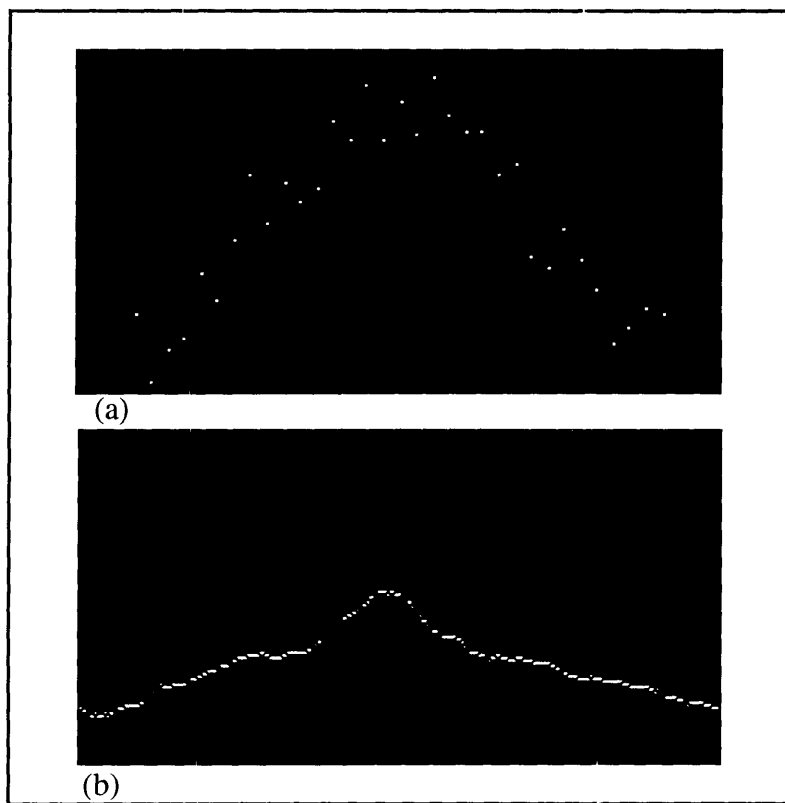
Figure 6.20 shows the settings for the `gen-noise-brownian` algorithm as used in mm.64-74 of *Descendant Lines*. As this is a vector generator, the processing function `vector-to-symbol` is defined first, allowing the mapping of `gen-noise-brownian` numeric values to the range of symbols a-h. The level setting of 5 (32-1) results in a sequence of 33 symbols, while the ‘smoothness’ setting of the random walk is set to a medium level of 0.5.

Figure 6.20 *Descendant Lines*, mm.64-74, `gen-noise-brownian` settings and output

```
(vector-to-symbol a h (gen-noise-brownian 5 0.3 0.5))  
output: c a b b c c d f e f e e g g h c g g h g g g f f d d d d c b b c c
```

Figure 6.21(a) shows the algorithm output in graphic form, as viewed in *Symbolic Composer's* Visualizer, the output of the algorithm clearly providing a mountain shape. Figure 6.21(b) presents the visual image produced following an increase in the level setting to 8 (producing 257 output values) and a maximum increase in the ‘smoothness’ setting to result in adjacent random walk steps. The resulting graphic models a portion of a mountain range.

Figure 6.21 *Descendant Lines*, gen-noise-brownian algorithm, Visualizer representations



In mm.64-74 of *Descendant Lines* the `gen-noise-brownian` algorithm is employed in the right-hand part of accordion 2. The output of the algorithm, shown in Figure 6.20, is modified with two *Symbolic Composer* processing algorithms `find-change` and `filter-preserve`. The `find-change` algorithm replaces repeated symbols with a rest symbol (=). The `filter-preserve` algorithm allows a range of symbols within a symbol pattern to be specified and retained, while remaining symbols are deleted. Figure 6.22 shows the resulting outputs of the two algorithms in relation to the output of `gen-noise-brownian`. In effect, the two processing algorithms reduce the output of the `gen-noise-brownian` algorithm to a symbol pattern that excludes repeated symbols.

Figure 6.22 *Descendant Lines*, mm.64-74, Processing of gen-noise-brownian output

```
gen-noise-brownian 5 0.3 0.5
output: c a b b c c d f e f e e g g h g g g h g g g f f d d d d c b b c c

find-change
output: c a b = c = d f e f e = g = h g = = h g = = f = d = = = c b = c =

filter-preserve'(a b c d e f g h)
output: c a b c d f e f e g h g h g f d c b c
```

The output of the `filter-preserve` algorithm is mapped to the non-transposed folksong note set throughout the mm.64 -74 sub-section, and results in a *locrian* modal melody that is accompanied by the set of three diminished chords used in the opening section of the work. The accompaniment provided by accordion 1 in this section is limited to percussive effects of the accordion. These are button tapping, aeolian bellows sounds and key taps, the rhythms for which are drawn from the folksong rhythmic cells, and are organised with symbol to rhythmic cell definitions. The symbols used for the part are the output of the `filter-preserve` algorithm as used for pitch in the accordion 2 part.

As the basis of the mm.64-74 subsection, the `gen-noise-brownian` algorithm, as a random walk process, serves to portray a programmatic journey through a natural landscape. The output symbols, mapped to the Croatian note set in *Descendant Lines*, function to portray the journey of the Croatian migrant.

The final chaos-based algorithm used in *Descendant Lines* is the `gen-lorenz` algorithm. This algorithm generates the Lorenz attractor, a model developed in 1963 by the meteorologist Edward Lorenz to describe the chaotic behaviour of the evolution of weather regimes in the atmosphere:

The atmosphere behaves like a turbulent fluid, and Lorenz was only too aware that it was governed by a set of mathematical equations that were non-linear and extremely sensitive to small changes — in other words, showed modes of instability. [Lorenz] sought a way of simplifying these equations so that he could study them mathematically, while retaining their essential non-linearity and instability (Palmer 1991, p.72).

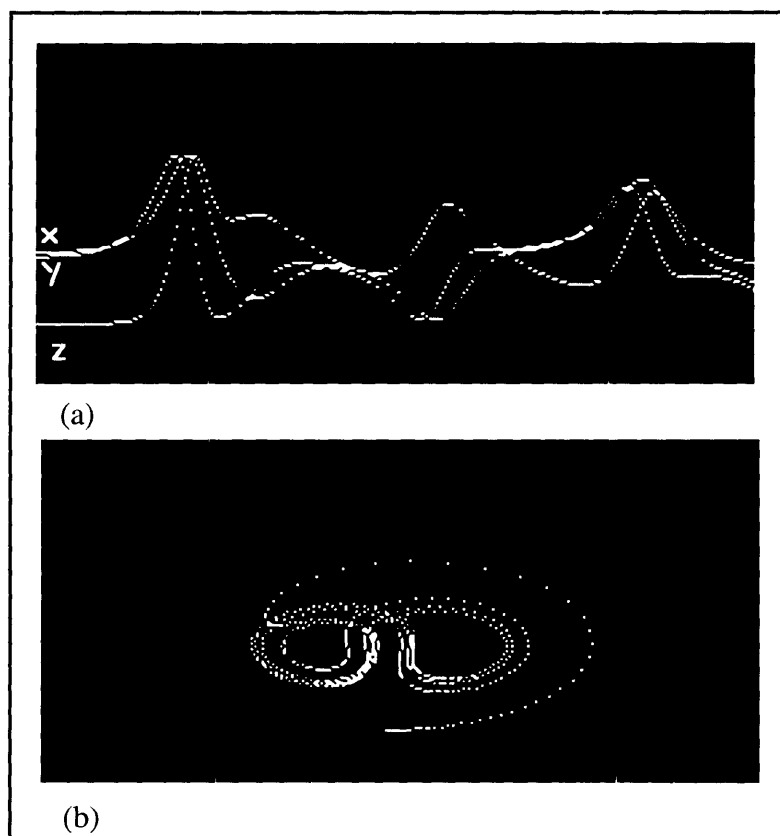
A simplified version of the equations led to a model with three variables, x , y and z :

A state of instantaneous weather in Lorenz's model can, therefore, be presented by a point in a three-dimensional 'phase space', and the evolution of the weather with time can be represented by a line, or trajectory, in this space. The climate of the model, the set of all possible weather states, is known as the Lorenz attractor (Palmer 1991, p.72).

The Lorenz attractor looks 'rather like a mask with two eyeholes. Trajectories wind round the two eyeholes of the mask, where both eyeholes merge together' (Stewart 1991, p.52). The eyeholes are in effect 'two lobes around which chaotic behaviour is attracted' (Thomas & Morgan 1990, p.229).

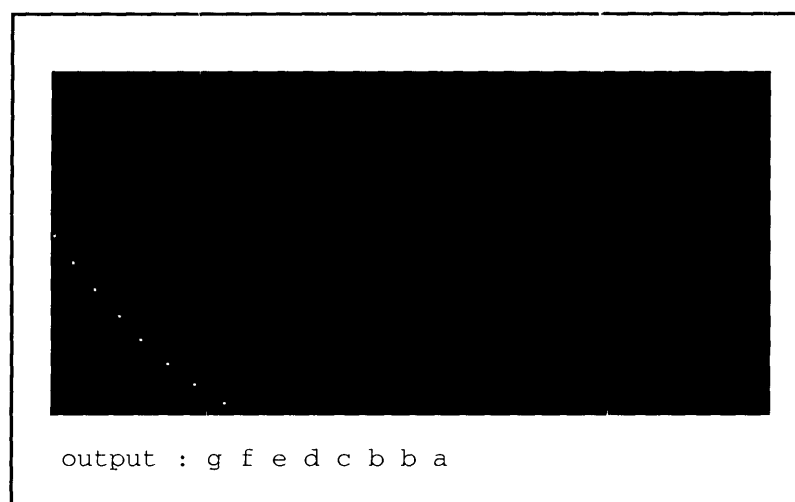
Figure 6.23(a) is a *Symbolic Composer Visualizer* representation of three individual trajectories x , y and z . When the three trajectories are applied to the `gen-lorenz` algorithm, their combined appearance results in the mask-like image of the Lorenz attractor, as shown in Figure 6.23(b).

Figure 6.23 *Symbolic Composer*, `gen-lorenz`., Visualizer representation



Within *Symbolic Composer*, the length of the trajectories for the `gen-lorenz` algorithm are controlled numerically. In the Figure 6.23 examples, the trajectory lengths were set to 600, in order to provide a representation of the mask-like shape of the Lorenz attractor. The numeric specifies the number of vectors generated by the algorithm, which in turn can be converted to symbols. In this case, 600 symbols would be generated. In mm.30-7 of *Descendant Lines*, a short, eight-symbol sequence was generated with the `gen-lorenz` algorithm, representing the starting point, or core, of chaotic movement producible with the Lorenz attractor. Figure 6.24 shows the Visualizer representation of the sequence and the resulting symbolic output where generated vectors are mapped to symbols in the range from a to g.

Figure 6.24 *Descendant Lines*, mm.30-7, `gen-lorenz` algorithm, Visualizer representation



In mm.30-7 of *Descendant Lines*, the `gen-lorenz` symbol sequence is used in combination with `gen-notrans a 2` based on the `tango defsym` definitions. The `symbol-transform` processing algorithm is used to transform pitch material generated by `gen-notrans` into pitch material mapped to the `gen-lorenz` sequence. Figure 6.25 shows the symbol output of `gen-notrans a 2` and `gen-lorenz`, the result of applying the `symbol-transform` algorithm, and the symbol to pitch mapping used. The `gen-lorenz` symbols are shown in bold type, the remaining symbols are from `gen-notrans`. Throughout

mm.30-7 the `symbol-transform` sequence is applied in the right-hand parts of both accordions.

Figure 6.25 *Descendant Lines*, mm.30-7, `symbol-transform` resulting output

gen-notrans a 2:	a	b	c	d	e	d	e	g
gen-lorenz:	g	f	e	d	c	b	b	a
symbol-transform:	g	b	c	d	e	d	e	g
	e	g	g	f	e	d	c	b
Pitch mapping (Tango note set)								
a	b	c	d	e	f	g		
A	Bb	C	D#	E	A	Bb		

Programmatically, the `gen-lorenz` sequence, as used in mm.30-7, represents the initial disintegration of the Argentinian's lifestyle, the Argentinian's identity previously established with `gen-notrans` in the immediately preceding mm.28-9. The `symbol-transform` process portrays the change from a secure environment to an environment developed with a chaotic seed.

The preceding descriptions of the applications of grammar-based and chaos-based algorithms are limited to examples where the discussed algorithms are used within *Descendant Lines*. Each algorithm is, however, generally used more than once within the work. Succeeding uses of individual algorithms are always in the same programmatic context as the initial use, with the exception of those used in the final section that function to provide a summation of the programmatic identities portrayed in the work. Table 6.3 shows the sections and subsections of the work and the corresponding algorithms used for generation of both pitch and rhythmic materials.

Table 6.3 *Descendant Lines*, Subsections and corresponding algorithms

(mm.#)	Program	Pitch	Rhythm
1-15	The Croatian	Folksong note set gen-notrans	Folksong cells (no generative algorithm)
16-18	Lifestyle disintegration	Folksong note set gen-hopalong	Folksong cells (no generative algorithm)
19-27	Escape	Folksong note set gen-hopalong, symbol- transform, gen- notrans	Folksong cells (no generative algorithm)
28-9	The Argentinian	Tango note set gen-notrans	Tango cells (no generative algorithm)
30-7	Lifestyle disintegration	Tango note set gen-notrans, symbol- transform, gen-lorenz	Tango cells (no generative algorithm)
38-53	Lifestyle Disintegration (cont.)	Tango note set gen-lorenz	Tango cells gen-lorenz
54-63	Departure		Folksong/Tango cells gen-notrans
64-74	Croatian journey	Folksong note set gen-noise-brownian	Folksong cells gen-noise-brownian gen-notrans
75-9	Argentinian's partner	(No generative algorithms)	Tango/Folksong cells gen-noise-brownian
80-4	Croatian's partner	(No generative algorithms)	Folksong/Tango cells gen-noise-brownian
85-94	Argentinian journey	Tango note set gen-noise-brownian	Tango cells gen-noise-brownian gen-notrans
95-108	Arrival	Tango note set gen-notrans	Tango/Folksong cells gen-notrans

Table 6.3 (cont.)

109-20	Argentinian growth	Tango note set gen-notrans	Tango cells gen-notrans
121	Argentinian son	Tango note set gen-fibonacci gen-notrans	Tango cells gen-notrans
122-3	Croatian growth	Folksong note set gen-notrans	Folksong cells gen-notrans
134	Croatian son	Folksong note set gen-rewrite	Folksong cells gen-rewrite
135-65	New society	Folksong/Tango note sets gen-rewrite gen-fibonacci gen-notrans	Folksong /Tango cells gen-rewrite gen-fibonacci gen-notrans

6.4. Editing

On completion of the initial *Symbolic Composer* scores for *Descendant Lines*, the scores were compiled and saved as MIDI files. In their MIDI file formats, the scores were transported to the *Vision* sequencing software program and were ‘auditioned’, the sequencing software allowing the playback of the compiled MIDI data through a sound module containing accordion timbres. When both *Symbolic Composer* and *Vision* are used simultaneously, aesthetically unsatisfactory output from a *Symbolic Composer* score can be heard with *Vision*, and parts of the score can be immediately revised. The revised score can then be compiled as another MIDI file and auditioned in *Vision*. In the development of the *Descendant Lines* MIDI data, this process was used extensively, and repeated until aesthetically satisfactory results were achieved from each *Symbolic Composer* score.

The final MIDI file for each subsection of *Descendant Lines* was then transported to the notation program *Finale*, where the raw MIDI data was transformed into a useable notation for accordionists and where dynamic levels, phrasing, and related score markings were added. Within *Finale*, editing of rhythmic materials was also carried out. As an

example, in the timesheet used for the opening measures of the work, there is a break of four 1/4 note lengths in mm.7-8 of the accordion 1 part, as shown in Figure 6.26. In the notated score, this period would normally contain rests, however in these measures a rest in the part would leave material presented in the accordion 2 part 'exposed', an undesired effect in this slow moving and texturally continuous subsection. In *Finale*, the rests that originally appeared in the accordion 1 part are replaced, the F that occurs on the third beat of measure 7 is extended through until the following G# of measure 8 that occurs with the next active area of the timesheet. Throughout *Descendant Lines*, such rhythmic editing occurs frequently, providing textural continuity within the work, and also provides variations to the limited number of tango and folksong-based rhythmic cells that are the primary basis of the work's rhythmic materials.

Figure 6.26 *Descendant Lines*, mm.7-8, rhythm editing

Measures 7 - 8 |---|---|

accordinrh

Original

Edited

6.5. Summary

The *Symbolic Composer* program offers the user a much larger array of algorithms than those shown in this study. As an example the `gen` prefixed (generative) algorithms alone number 35 in the version of the program used for this study. However, the relatively

few algorithms presented in the above examinations are representative of a range of the program's capabilities, and demonstrate some of the program's mathematically and scientifically based processes that generate markedly different pitch and rhythmic sequences on limited initial materials.

The *Symbolic Composer* algorithms used in *Descendant Lines* provide a most adequate vehicle for the generation of pitch and rhythmic materials, while the inherent nature of each algorithm, either grammar-based or chaos-based, is equated to programmatically portrayed events within the work. The grammar-based algorithms serve to build the pitch and/or rhythmic identities of the characters portrayed, while the chaos-based algorithms serve to portray the more turbulent times in the lives of the Croatian and Argentinian migrants with whom the work is programmatically concerned.