Grammar Based Music Composition

Jon McCormack
Computer Science Department
Monash University, Clayton Victoria 3168
email: jonmc@cs.monash.edu.au

Abstract

L-Systems have traditionally been used as a popular method for the modelling of space-filling curves, biological systems and morphogenesis. In this paper, we adapt string re-writing grammars based on L-Systems into a system for music composition. Representation of pitch, duration and timbre are encoded as grammar symbols, upon which a series of re-writing rules are applied. Parametric extensions to the grammar allow the specification of continuous data for the purposes of modulation and control. Such continuous data is also under control of the grammar. Using non-deterministic grammars with context sensitivity allows the simulation of Nth-order Markov models with a more economical representation than transition matrices and greater flexibility than previous composition models based on finite state automata or Petri nets. Using symbols in the grammar to represent relationships between notes, (rather than absolute notes) in combination with a hierarchical grammar representation, permits the emergence of complex music compositions from a relatively simple grammars.

McCormack, J.: Grammar Based Music Composition, in Complex Systems 96: From Local Interactions to Global Phenomena, R. Stocker et. al., eds, ISO Press, Amsterdam, pp. 320-336, 1996.

1. Introduction

Music is one of those areas of activity that, despite its ubiquitous presence in human culture, remains largely immune to a detailed understanding. Nobody knows why something so apparently simple as the succession of changing discrete tones has the power to move the listener emotionally. Of all the arts, music is considered "to be open to the purest expression of order and proportion, unencumbered as it is by material media" [16]. It has been often noted that music bears a close affinity to mathematics, and that notions of mathematical and musical aesthetics may have some similarities [31]. In recent years, many researchers and musicians have turned to the computer as both a compositional, and synthesis device for musical expression.

In this paper we describe a system for computer assisted music composition and thus the majority of this introduction will focus on the application of computers for composition. This is not to discount the many other uses of computers in music, such as sound synthesis, acoustic and physical modelling, automated notation, score editing and typography.

A major part of music creation involves the use of certain musical *formalisms* (systematic ordering), as well as algorithmic and methodic practices. Many of these were well established before the advent of digital computers. In a substantial survey of computer composition, Loy [15] suggests that the application of musical formalisms based on a priori theories of composition have been largely developed this century, originating with composers such as Schoenberg and Hindemith. Loy also notes the contribution of music typographers and instructors who have found formal systems attractive to their profession.

Representations for notation and programming of music (and thus computer composition) can be broadly classified into *symbolic* and *iconic* [16]. Symbolic representations relate symbol and a sonic event, but have no direct resemblance between them. Iconic representations carry some resemblance (usually visual) to the sound the icon represents. Common music notation combines both forms of representation (time signature, cleffs, bar lines are symbolic; pitch coding, crescendo, diminuendo are iconic). While this system is not completely adequate, particularly for some contemporary forms of music, it is an extremely versatile and capable notation for a diverse range of musical expression.

1.1 Musical Patterns

It has been observed that almost all forms of music involve repetition [13], either of individual sequences of notes or at some higher levels of structural grouping. Often these repetitions appear at a number of different levels simultaneously. Some compositions repeat patters that are slightly changed at each repetition, or revolve around some musical 'theme', whereby complex harmonic, timing or key shifts in a basic theme provide a pleasing musical diversity.

It is also commonly known that what allows us to identify an individual piece of music is the *change* in pitch between notes, not the pitch of the notes themselves. We can change the key of a composition (equivalent to multiplying all frequencies by some fixed amount) and still recognise the melody.

A wide variety of approaches have been taken to the compositional problem. In Loy's survey paper he details many different methods: stochastic and combinatorial models; grammar based; algorithmic; process models; and other models derived from Artificial Intelligence techniques.

The methods used in the system described in this paper are grammar based. While grammars have been a popular method for algorithmic composition, in most instances the grammars used are relatively primitive and thus limit the scope of control and diversity of composition that can be generated. The approach used in the system described in this paper, is to take recent developments in grammars used for modelling of biological morphogenesis and herbaceous plants in computer graphics, and apply them to music composition. In addition to being able to represent a variety of both stochastic and deterministic compositional techniques, the system has compositional properties unique to this approach.

The remainder of this paper is organised thus: Section 2 briefly looks at the problem of computer based musical creativity and explains the rationale behind *computer assisted composition*. Section 3 examines algorithmic representations for music composition and introduces the basis of the grammar method. Section 4 details how grammars can be adapted for music composition, with specific emphasis on Lindenmayer Systems (L–Systems) as the basis for the model. Section 5 discusses implementation details. Finally, Section 6 details possible extensions and further work.

2. Creativity

A complete theory of general creativity, or even musical creativity, remains elusive. While many attempts have been made to study and document the creative process: [1, 5, 31, 10] for example, any generality seems difficult to uncover. In many cases people do not know how or why they make creative decisions, and much of the creative process is difficult to repeat in controlled experiments.

The question of whether creativity is computable is an issue of even greater controversy. Many researchers from Poincaré [25] to Penrose [24] have argued against a computable model of creativity because, simply put, the underlying mental processes are not computable. These arguments depend largely on speculation and the personal opinion of the authors (although Penrose does base his objections on an as yet untested theory of the relationship between consciousness and Quantum theory). It is this author's opinion that while creativity is clearly an ability of the human mind, to a large extent practical creativity is deeply related to an individual's life experience. A life experience includes relationships to the environment, interaction with both living and non-living things, social and cultural constructions. We all know that these things have a major effect on a person's internal states. Much creativity also depends on serendipitious and chance events in the external world, both conscious and unconscious. The explicit use of random events in composition is a practice many centuries old. It can be found in the works of many artists – from Mozart and Hayden, to Burroughs and Pollack. Creativity is not only influenced by external events, it may also be stimulated by artificially induced internal effects – some artists claim their best creative work is done under the influence of a large variety of chemical substances.

Some day it *may* be possible to build a computer with processing capabilities similar to that of the human brain [6], however computing a simulated life experience would appear to be impossible [22]. It is feasible that a robot that has life-experience in the real world may have the potential to exhibit creative behaviour, but again if life-experience is bound to physicality and matter (as opposed to mechanisms and processes) any creativity exhibited by such a robot may not be recognised as human-like creativity, or even recognised as creativity at all.

2.1 Semantics and Meaning

Johnson–Laird [10], has argued that music is an excellent testing ground for theories of creativity, because, as opposed to other Artificial Intelligence domains such as natural language generation, music avoids the problem of semantics. That is, a musical expression cannot be judged to be true or false. While this statement is literally correct, many musicians would argue that as part of the compositional process, musical expressions *do* contain semantics. For example, some composers associate emotive, structural or linguistic semantics with individual themes or passages in a composition. Carl Orff described musical composition as the raw expression of human energies and states of being [23]. The musical development of a composition is driven by the associated development of emotions, structure or language. In this sense such compositions do have 'meaning' and there is a semantic association between musical expressions. Such expressions have ordering and denote an emotive symbolic system. According to Langer [12], music, as a symbolic system of emotional archetypes, conveys a "morphology of feeling", akin to algebraic notation conveying a mathematical expression.

Some composers (Brian Eno, for example) visualise abstract or real environments as mental images, and the visual and emotive feeling of these environments directly influences the composition. Composition is rarely a one way process from mind to finished composition either. It is an *iterative feedback* process – hearing the music causes change in the composition. Composers are rarely interested in one aspect of the composition in isolation, such as pitch, timing or timbre; rather these things in total – the 'surface' of the music – greatly influence the emotional feeling and thus carry the composition.

In summary, while music may appear to be an easier domain to test theories of creativity, it is difficult to expect a computer program to exhibit a genuine musical creativity which approaches that of human composers. A more achievable approach, one adopted by the system described in this paper, is to use the machine as a synergetic partner to the human composer. The human and computer work in tandem through an interactive feedback process, the computer presenting new musical possibilities by synthesising complexity and variation, the human composer directing the overall creative 'quality' of the composition. This human-machine feedback process, known as *computational synergetics* has been used to significantly enhance both scientific [33] and artistic [20] creativity. The computer, used in heuristic mode can lead the human collaborator to new discoveries that may have been difficult or impossible without the collaboration. In this way, the machine acts as a kind of 'creative amplifier', enhancing the creative potential of the composer.

3. Representations for Music Composition

3.1 Procedural Models

A grammar based approach to composition makes heavy use of *procedural models*. One advantage of procedural methods in general, is that they allow a terse representation of a complex model. This is often referred to as *database amplification*. Such techniques are not the exclusive domain of computing. The representation of a complex organism is effectively encoded in the much simpler (in relative terms) DNA. Likewise with grammars, there is the possibility for the *emergence* of complexity through the repeated production of simple (and possibly deterministic) rules. One has to be careful with such analogies however, for biological morphogenesis relies also on many factors, such as the laws of physics, special properties of carbon atoms, the *self-*

organisational properties of many molecular and cellular structures and complex environmental interactions. Certainly, these features are not built into our music compositional model, though it is possible that some of these concepts, in an abstract sense, could be incorporated into the model.

3.2 Stochastic Processes

A diverse variety of *stochastic* processes have been applied to music composition [11, 21 (chapter 5)]. Most involve a two stages. First, some existing musical expression or quantity is analysed. Following the analysis, re–synthesis is applied using the selected stochastic technique. Importantly, while we can undertake statistical analysis of many musical patterns, when we attempt to synthesise music based on that analysis, the results rarely seem to have the clarity or intent of human composition. For example, Voss [32] analysed many different types of music and found that several were statistically similar to *1/f* noise, however when one converts *1/f* noise to musical notes, the result have little in common musically with any of the compositions from which the statistical analysis was derived. Given the discussion in Section 1.1 regarding formalisms, it is clear that a more sophisticated model is required.

Generally in composition, one primary concern is the notion of temporal *events* and *event spaces*. We use the general term *event* to represent some basic building block of composition, in many instances a note or set of notes, but in other compositional applications events may refer to sound complexes, individual sound samples or other basic musical elements. An event space is an ordered set of events. The number of events contained in a given event space represents its *order*.

3.2.1 Markov Models

One popular approach to computer composition is the use of *Markov chains*, where the probability of a specific event *i,* is context dependent on the occurrence of a previous event or event space. Most attempts using this form of modelling first require some existing composition to be analysed. For the purposes of explanation, we will restrict this discussion to consider only the pitch of notes, however there is no reason why the same techniques cannot be applied to other musical qualities such as duration, volume or timbre. We also consider only *discrete–valued* (as opposed to *continuous–valued*) data. Markov processes are suited to the analysis of conventional musical pitches. A Markov process is considered *stationary* if the transition probabilities remain static.

An *N*–order Markov model can be represented using an *N+1* dimensional *transition matrix*. For example, Figure 1 shows a sequence of note events, and the corresponding transition matrix. This is a 1st–order model, where the probability of a given event depends only on the event immediately preceding it. Each element in the transition matrix, P_{ij} represents the probability of event *j* occurring given that event *i* has just occurred. *Pij* is calculated by summing the occurrences of each note *j* that follows note *i* in the input melody. Counts in each row are normalised so the combined probabilities for each row sum to 1, ie.:

$$\sum_{i=1}^{N} P_{ij} = 1 \ \forall_j \in [1..N]$$

Set of input notes:

E D C D E E E D D D E G G E D C D E E E E E D D E D C

Transition Probability Matrix:

Current Event \ Next Event	C	D	E	F	G
C		2/3	1/3		
D	3/10	3/10	4/10		
E		5/11	5/11		1/11
F					
G			1/2		1/2

Figure 1: A set of input notes (from the nursery rhyme "Mary had a little lamb"), and the corresponding 1st order Markov model, represented as a transition matrix. Each element of the table P_{ij} is the probability of event j given that event i just occurred. Empty elements in the matrix represent a probability of 0.

Jones [11] shows how transition tables can be converted to *event–relation* diagrams, which are essentially finite-state automata. Lyon [17] has used Petri Nets to extend the finite-state model generated by stationary *N*–order Markov models for real–time performance.

The Markov process for composition has several problems:

- Unless the transition table is filled with randomly generated probabilities, some existing music must be input into the system. The resulting Markov model will only generate music of similar style to the input.

- For higher order models, transition tables become unmanageably large for the average computer. While may techniques exist for a more compact representation of sparse matrices (which usually result for higher order models), these require extra computational effort that can hinder real-time performance.

- Most importantly, while such models have been used successfully in composition, they are limited in the variety of music produced by any given transition table. They also provide little support for structure at higher levels (unless multiple layered models are used where each event represents an entire Markov model in itself).

3.3 Grammar Based Approaches

The grammar based approach is not new to music composition. Early users of grammar based techniques include Buxton et. al [2], and Roads [30]. Holtzman's Generative Grammar Definition Language (GGDL) compiler [8, 9] is based on Chomsky grammars of type 0 to 3 (regular, context-sensitive and context free) [3, 4]. Jones [11] discusses context–free *space grammars* that can operate across many dimensions, where each dimension represents a different musical attribute. An interesting property of space grammars is that an *n*–dimensional grammar can generate an *n*–dimensional geometric shape, which may provide a higher level iconic representation of the music generated by a particular set of rules.

Little use of L-Systems has been made in the area of music composition. Prusinkiewicz [29] has used two–dimensional space filling curves generated by L–Systems as a basis for music generation, where the spacial position of a vertex in the curve represents a note, the edge length the duration.

Much development in music composition and synthesis has been performed in closely related areas such as fractal and stochastic methods, procedure-oriented and recursive composition. Moore [21] gives an overview of some of these methods.

In techniques that use Chomsky grammars, the rewriting is *sequential*, that is, symbols are replaced one at a time sequentially across the string (corresponding to the serial nature of the process that the grammar represents). In the system described in this paper we adapt *parallel* rewriting grammars for composition. Due to constraints discussed in Section 4.2, rewriting is not fully parallel as is the case with L–Systems (upon which the system is based). Rewriting proceeds on parallel sets of elements in the string after the events represented by the string have occurred.

4. L-Systems for Music Composition

4.1 L–Systems

L-Systems are a form of string rewriting grammar, first developed by Lindenmayer in 1968 [14]. The original emphasis was as a mathematical model of cell development and plant topology. Prusinkiewicz [26] applied a *turtle interpretation* to produced strings that create realistic models of plants for computer graphics purposes. Hanan developed *parametric* L-Systems that allow symbols in the grammar to have associated numerical parameters [7]. Work by Prusinkiewicz, et. al [27], developed L-Systems further, incorporating more complex models for growth (*differential* L-Systems) [28], geometric control and interaction with the environment. Hierarchical L-Systems, explained in Section 4.5, allow more complex sequences of rules and give the user greater potential for modelling of interacting event spaces at different levels. L-Systems have been used extensively in computer graphics for the visual modelling of herbaceous (non-woody) plants [27]. They have also been successful for a number of other applications including the general modelling of natural forms, morphogenesis, legged gaits of animals, image processing and the modelling of human organs for medical analysis.

L-Systems provide a compact way of representing complex patterns that have some degree of repetition, self-similarity or developmental complexity. Because of the abstract nature of the grammar itself, produced symbols have found a diversity of physical and structural representation. In the case of geometric modelling, L-Systems fundamentally provide a way of representing topology. Extensions to the basic grammar

allow the interpretation of the topology through complex environmental and structural parameters.

4.2 DOL–Systems

A basic introduction to L–Systems is presented here, for detailed explanations, the reader is referred to [27].

A deterministic, context–free L–grammar (DOL–System) is an ordered triplet:

$$\mathbf{G} = <\sum, P, \alpha>$$

where:

$\sum$ – alphabet of the system, $\sum^*$ the set of all words over $\sum$, $\sum^+$ the set of all non-empty words over $\sum$.

P – finite set of *productions* or *rules*.
A production $(a, \chi) \in P$ is written as: $a \rightarrow \chi$, where $a \in \sum$ and $\chi \in \sum^*$. a is termed the *predecessor* and χ the *successor* of the production.

α – a non-empty word called an axiom, $\alpha \in \sum^+$.

If, for any letter in the alphabet $a \in \sum$ there is no production, then the *identity production*, $a \rightarrow a$, is assumed (and included by default in the set of productions). Beginning with the axiom, α, we apply the production in parallel to all letters (symbols) in the word. This iterative process proceeds for a given number of levels. The resultant string is then interpreted as musical information.

For example, given the set of productions and an axiom:

p_1: C → E
p_2: E → C G C
p_3: G → ε (ε is an empty word or null)
α : C

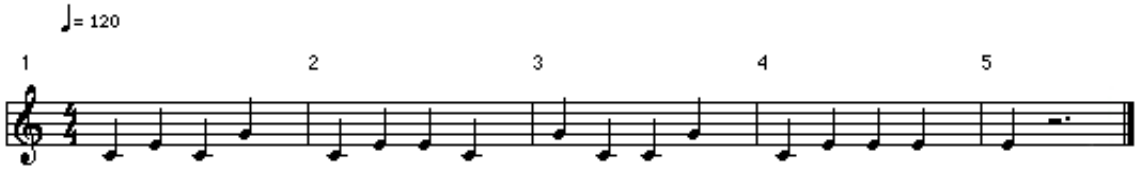
The produced string for the first five iterations becomes:

ITERATION	STRING
0	C (axiom)
1	E
2	C G C
3	E E
4	C G C C G C
5	E E E E

Concatenating the strings together gives:

C E C G C E E C G C C G C E E E E

If each letter is interpreted as a musical note, the string can be played as music:



This simple example has no provision for note timing, so each note defaults to a quarter–note duration. It is of course possible to incorporate symbols in the grammar that control duration.

4.3 Stochastic Grammars

Stochastic grammars directly map a set of probabilities, [p_{a1}, p_{a2}, ... p_{aN}] , with a set of the productions, [P_{a1}, P_{a2}, ... P_{aN}] involving a letter $a \in \sum$ as the predecessor. The probability is usually written above the arrow symbol. For example the production:

$$p_1\text{: } a \xrightarrow{1/2} b$$
$$\xrightarrow{1/2} c$$

Means that a has equal probability of being replaced by b or c. It is assumed that, for all productions with a as the predecessor, that the combined probabilities for that letter sum to 1, ie.:

$$\sum_{i=1}^{N} p_{ai} = 1$$

Stochastic grammars allow the representation of Markov models. For example the transition matrix shown in figure 1 is equivalent to the stochastic grammar:

p_1: C $\xrightarrow{2/3}$ D
$\xrightarrow{1/3}$ E

p_2: D $\xrightarrow{3/10}$ C
$\xrightarrow{3/10}$ D
$\xrightarrow{4/10}$ E

p_3: E $\xrightarrow{5/11}$ D
$\xrightarrow{5/11}$ E
$\xrightarrow{1/11}$ G

p_4: G $\xrightarrow{1/2}$ E
$\xrightarrow{1/2}$ G

α: E

The choice of axiom is arbitrary. For a correct simulation of the Markov chain the axiom should select a starting letter based on the frequency a note appears in the set of input notes (this can be incorporated into the grammar as one additional production).

4.4 Parametric Extensions

Numerical parameters may be associated with letters in the alphabet [7, 26]. Successor words in a production may involve mathematical expressions in the calculation of parameters. An arbitrary number of real–valued parameters may be associated with any letter. In addition, conditional statements resolve the application of rules based on parameter values:

$$A(x) : x < 64 \rightarrow A(x + 1)$$

In the example above, the production is applied while the parameter of A is less than 64. If the parameter in this case is interpreted as note volume, then the above rule plays a crescendo.

The following parametric grammar computes the Fibonacci series:

$$p_1 : A(x, y) \rightarrow A(y, y + x)$$
$$\alpha : A(1,1)$$
$$A(1,1) \Rightarrow A(1,2) \Rightarrow A(2,3) \Rightarrow A(3,5)...$$

In a musical application, parameters can be used to control continuous–valued attributes such as note velocity. The MIDI† specification allows for a number of continuous–valued controllers. These controllers can be effected through parametric association with the appropriate symbol.

4.5 Hierarchical Grammars

As stated, many forms of music involve repeating patterns, often at a number of different levels in a composition. While it is possible to incorporate many levels in a set of productions, a naturally easier way is to impose a hierarchical containment of rules in order to better represent the structure of the music.

Hierarchical grammars have a similar specification as a DOL–System grammar, except that on the successor side of a production, one or more of the successor symbols may be an entire grammar in itself. Development of each set of rules proceeds independently, however rule sets may pass parameters between sets of grammars. Figure 2 shows an example grammar and the resulting strings produced from the rules.

```
A = {
        B(x) = {
                p_b1 :   a(y) → a(y+1) c(y/2)
                α:  a(x)
        }
        p_a1 :    B(x) → a(x) [ B(2x) a(x) ]
        α: B(1)
}
```

† Musical Instrument Digital Interface (MIDI) is a standard method of control in electronic music. It allows the connection of various types of computers, synthesisers and keyboard controllers, provided they obey the MIDI protocol.

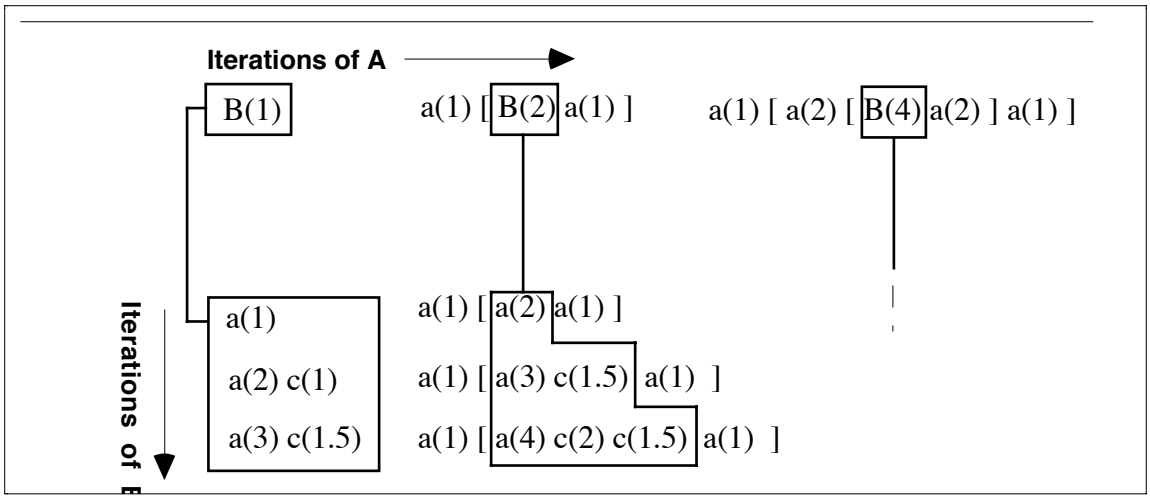


Figure 2: A simple hierarchical grammar consisting of two rule sets, **A** and **B**. Below the rules the diagram shows the produced strings, with iterations over system **A** running horizontally and iterations over system **B** running vertically (the iterations of **B** for the third iteration of **A** is not shown). Each system's development proceeds independently, but they may communicate via parameters.

5. Implementation

Thus far, we have introduced various types of grammars and listed some simple interpretation of the strings they produce for musical purposes. In this section we discuss the implementation of a composition system based on the grammars described in the previous section.

The system is implemented on a Silicon Graphics workstation connected to a digital synthesiser via MIDI. Once a grammar is parsed it is capable of generating note sequences (MIDI data) in real–time. Very complex grammars (such as those with many levels of hierarchy, or productions that cause exponential growth in the produced string) may cause noticeable delays in the system during processing. In this case sequences can be saved onto disk for latter playback. The goal of the system is real–time performance and in most cases this is adequately accommodated.

5.1 Data Flow

Essentially, the composition system has three major processing stages, outlined in Figure 3. The system builds lists of strings. Productions are applied to these strings and the results interpreted as commands that are output as MIDI data to various synthesisers connected to the system. MIDI data can be converted to standard notation, suitable for interpretation by real musicians (as opposed to machines) if required.

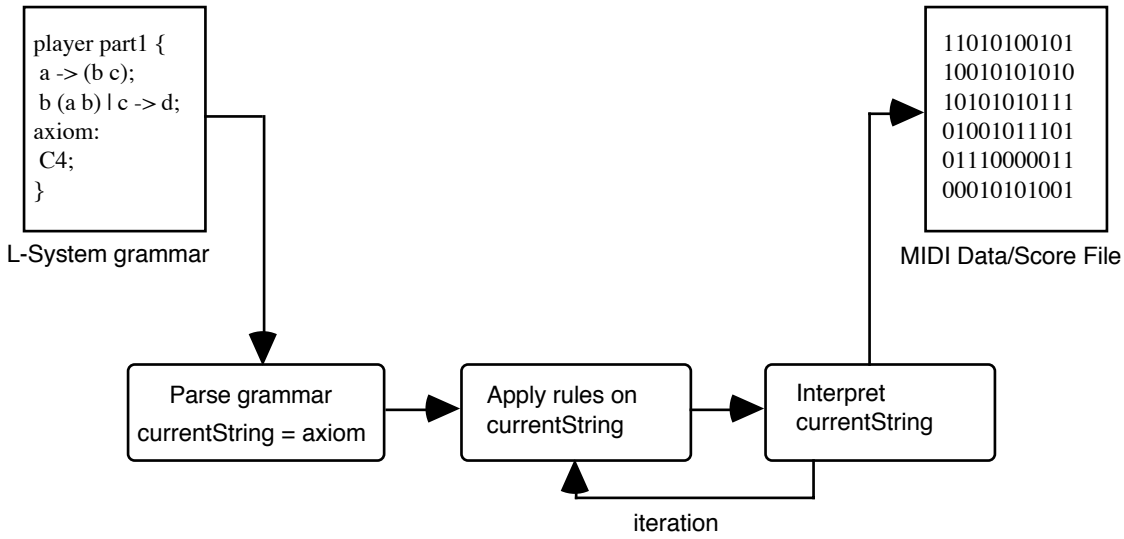


Figure 3: Stages of processing in the system. Square boxes represent data, rounded boxes represent processes. The lines with arrows show the directional flow of information. First a grammar is parsed, the axiom loaded into the variable `currentString`. Rule application is iterative. After rewriting is applied to `currentString`, the results are interpreted. The interpretation converts symbols into MIDI data that is then played, or saved to a file.

5.2 Players and Symbol Interpretation:

The system uses the notion of a virtual *player*. The player is responsible for the interpretation of instructions given to it, which in the general case means playing notes (sending MIDI messages or writing score information) Players always maintain a current *state,* that consists of:

- pitch (octave and note within current octave)
- duration
- timbre
- control parameters

Control parameters may be used to specify continuous or discrete controllers (eg. pitch bend, filter resonance, etc.).

Symbols in the produced string change the players state. The string is read sequentially, from left to right and each symbol is interpreted by the player, just like a Turing machine reads instructions from a tape. Different symbols cause different components of the player to change state, or perform a task. For example, the dot ('.') symbol tells the player to play the current note(s), a '+' increments the current note by a semitone, a '–' decreases it by the same amount. Players are capable of playing any number of notes simultaneously, however in practice this may be limited by output hardware. The square parenthesis symbols ('[' and ']') push and pop the current player's state onto a First–In, Last–Out stack.

Multiple players perform a *composition*. Any number of players may be involved in a composition, however again hardware limitations may fix an upper limit on the total number of notes playable simultaneously. Generally, each player plays a single voice or instrument and each has a different set of rules.

Individual notes can be represented by their name. The '#' symbol represents a sharp, '&' a flat and '@' a natural. Notes in uppercase change the pitch and play the

note. Notes in lowercase change the pitch but don't play the note. This means that 'C' is equivalent to 'c .'. A comma ',' plays a rest for the current duration. Other symbols change duration, timbre, and so on.

Polyphonic playing of notes is achieved by enclosing the notes to be played simultaneously in round parenthesis: '(' and ')'. For example the string:

C E G

plays the specified notes, each for the current duration, in temporal sequence from left to right. The string:

(C E G)

plays the same three notes, for the current duration, simultaneously (a C Major chord). Pitch specification can be relative, thus:

c (. + + + . + + + .)

Does the same thing (plays a C Major chord).

5.3 Polyphony and Context

Context sensitivity relates production application to the predecessor's context (symbols that appear before the predecessor in an axiom or string). In the case of music composition, context sensitivity is represented in the grammar both polyphonically (notes currently being played) and temporally (note(s) previously played). A vertical bar ('I') is used to signify temporal context: symbols to the left of the bar must precede the current symbol for the rule to apply. For example:

String	Interpretation
(C E G) → (G B D)	Polyphonic context: if current notes are C, E and G play G B and D simultaneously.
C E I G → D	Temporal context: if current note is a G preceded by E and C then play a D.
(C E) I (G C) → D (C E)	Polyphonic and temporal context: if current notes are G and C and they were preceded by C and E played simultaneously then play D for the current duration followed by C and E simultaneously.

The use of context in a grammar allows for complex shifting themes to propagate through a composition, both chromatically and temporally. Context is necessary for a grammar to implement Markov models of order 2 or more.

6. Conclusions and Future Work:

Development of the system is still at early stages, but already it is possible to see the potential of the grammar based approach. The use of stochastic grammars allows the compact representation of Markov chains. The parallel rewriting technique of L–grammars has capabilities equal to, or better than, those of previous grammar based

methods. Parametric numerical parameters allow the control of non–discrete processes such as attack velocity or volume. Finally, hierarchical and context sensitive grammars allow the simultaneous development of complex patterns at different levels within a composition.

In the introduction, it was emphasised that this is a computer assisted composition system, where composer and machine work in a synergetic tandem. Currently this tandem is little better than the way a programmer edits and debugs source code. Grammars must be written, compiled (removing syntax errors in the process) before music can be heard. Currently work is underway to make the system much more interactive from a performance point of view. A composer will author a grammar, and within the grammar place areas of discrete and continuous control that can be influenced by external processes or systems. One such example would be the recognition of hand gestures (gestures have both continuous and discrete information), in the spirit of the GROOVE system of Mathews and Moore [18]. In this way the composer can act as conductor, instantiating medium–term changes in the performance through gesture (discrete gestures cause rule changes) and also influencing the short term quality of the performance (continuous data from hand positions in space). Development of such a system is currently underway.

It is also possible to apply 'genetic' processes to grammars; to interactively evolve better sounding compositions using a novel variation of the genetic algorithm. Evolution of grammars has already been done for computer graphics modelling [19]. Starting with a base grammar, a number of different mutated versions of the grammar can be represented as pictorial icons on the computer screen. By moving the mouse towards an icon the contribution to the composition is increasingly biased towards the associated grammar. If the mouse is completely on the icon, only that grammar is heard. Once the user finds the best spacial position (and thus the best sounding composition) the selected grammars are 'mated' and become the new parent grammar. The process is repeated for as long as the user wants, hopefully evolving the grammar into a better sounding composition.

While this technique sounds appealing, there is a limit to the number of mutations that can be displayed on the screen at any one time and also to the number of generations that can be critically evaluated by a human during any one sitting. We are still a long way from allowing the machine to exhibit creative judgement on its own.

7. References

1. Barron, F. *Creative Person and Creative Process,* Holt, Rinehart and Winston, Inc. (1969).

2. Buxton, W., Reeves, W., Baeker, R., and Mezei, L. The Use of Hierarchy and Instance in a Data Structure for Computer Music. *Computer Music Journal 2*, 2 (1978), 10 – 20.

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

4. Chomsky, N. Formal Properties of Grammars. In *Handbook of Mathematical Psychology, Volume II*. Wiley, Luce, R., Bush, R., and Galanter, E., New York, 1963.

5. Crutchfield, R.S. The Creative Process. In *Creativity: Theory and Research*. College and University Press, Bloomberg, M., pp. 54 – 74, , 1973.

6. H, M. *Mind Children: The Future of Robot and Human Intelligence,* Harvard University Press, Cambridge (1988).

7. Hanan, J. *Parametric L–Systems and their Application to the Modelling and Visualisation of Plants*, Ph.D. dissertation, University of Regina, June 1992.

8. Holtzman, S.R. A Generative Grammar Definition Language for Music. *Interface 9*, 2 (1980), 1 – 48.

9. Holtzman, S.R. Using Generative Grammars for Music Composition. *Computer Music Journal 5*, 1 (1981), 51 – 64.

10. Johnson–Laird, P.N. *Human and Machine Thinking,* Lawerance Eribaum Associates (1993).

11. Jones, K. Compositional Applications of Stochastic Processes. *Computer Music Journal 5*, 2 (1981), 381 – 397.

12. Langer, S. *Philosophy in a New Key,* New American Library, New York (1948).

13. Leach, J. and Fitch, J. Nature, Music, and Algorithmic Composition. *Computer Music Journal 19*, 2 (1995), 23 – 33.

14. Lindenmayer, A. Mathematical Models for Cellular Interaction in Development, Parts I and II. *Journal of Theoretical Biology 16*(1968), 280 – 315.

15. Loy, G. and Abbott, C. Programming Languages for Computer Music Synthesis, Performance and Composition. *ACM Computing Surveys 17*, 2 (1985).

16. Loy, G. *Composing with Computers – a Survey of Some Compositional Formalisms and Music Programming Languages,* MIT Press: Cambridge, Massachusetts (1989)pp. , 291 – 396, Chapter 21.

17. Lyon, D. Using Stochastic Petri Nets for Real–Time Nth–order Stochastic Composition. *Computer Music Journal 19*, 4 (1995), 13 – 22.

18. Mathews, M.V. and Moore, F.R. GROOVE: A Program to Compose, Store and Edit Functions of Time. *Comm. ACM 13*(1971), 715 – 721.

19. McCormack, J. Interactive Evolution of L–System Grammars for Computer Graphics Modelling. In *Complex Systems: from Biology to Computation*. ISO Press, Green, D. and Bossomaier, T., pp. 118 – 130, Amsterdam, 1993.

20. McCormack, J. Wild: An Interactive Computer Installation. In *The 1994 Next Wave Art and Technology Catalogue*. Next Wave Festival, Inc., Sproul, L., pp. 22 – 25, Melbourne, 1994.

21. Moore, F.R. *Elements of Computer Music,* Prentice–Hall, Englewood Cliffs, New Jersey (1990).

22. Oreskes, N., Shrader–Frechette, K., and Belitz, K. Verification, Validation and Confirmation of Numerical Models in the Earth Sciences. *Science 263*(February 1994), 641 – 646.

23. Orff, C. *Gesspräche Mit Komponisten,* Züric (1967).

24. Penrose, R. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics,* Oxford University Press (1989).

25. Poincaré, H. *The Foundations of Science: Science and Hypothesis, the value of Science, Science and Method,* The Science Press, New York (1923).

26. Prusinkiewicz, P., Lindenmayer, A., and Hanan, J. Developmental Models of Herbaceous Plants for Computer Imagery Purposes. In *Computer Graphics,* Proceedings of SIGGRAPH 88 (Atlanta, Georgia, August 1–5, 1988), ACM SIGGRAPH, New York, 1988, pp. 141 – 150.

27. Prusinkiewicz, P. and Lindenmayer, A. *The Algorithmic Beauty of Plants,* Springer–Verlag, New York (1990).

28. Prusinkiewicz, P., Hammel, M.S., and Mjolsness, E. Animation of Plant Development. In *Computer Graphics,* Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993, ACM SIGGRAPH, New York, 1993, pp. 351 – 360.

29. Prusinkiewicz, P., " ," 1996, Personal Communication.

30. Roads, C. Grammars as Representations for Music. *Computer Music Journal 3*, 1 (1979), 45 – 55.

31. Schillinger, J. *The Mathematical Basis of the Arts,* The Philosophical Library, New York (1948).

32. Voss, R.F. and Clarke, J. 1/F Noise in Music and Speech. *Nature 258*(1975), 317 – 318.

33. Zabusky, N.J. Computational Synergetics. *Physics Today* (July 1984), 36 – 46.