

Investigation of AI Computer Music Composition Techniques

Jeffrey Grafton

January 23, 2004

Abstract

Computer composition of music through artificial intelligence techniques is a field that has seen research since the early days of computers. This project aims to investigate some of these algorithms, specifically using Markov chains and Lindemayer systems, then hopes to improve on them, producing higher quality music. A common question is whether it is likely to ever produce a model or algorithm that can creatively write music, as it is something that is hardly understood by humans. This project will not focus on developing creativity, but will hope to produce a better end product.

1 Background

Compositional techniques have been researched since the earliest days of the computer, though at the time, were often very primitive and mechanical. Various models, grammars, and algorithms have been produced since that time, all of which can imitate previously written human compositions or produce new, undirected music. Recent research has provided models that allow more unique results, but the products are still constrained by the rules set by the programmer.

2 ABC Plus Music Notation

The ABC Plus music notation language is a highly portable ASCII-based notation format. ABC was originally designed for only melody; it did not support more than one line of music simultaneously. The ABC Plus project aims to extend the ABC format, allowing for polyphonic music. Not all sites are using the same implementation, however, so there are some incompatibilities that exist.

For the purposes of this project, a single melody will be used initially, extending to polyphonic music later in development. The program will output an ABC format file which can easily be converted to MIDI for playback on the computer speaker, or it can be converted to a Postscript file, which can be sent to the printer and performed by humans.

3 Markov Chains

Markov chains provide a way to model the likelihood of events following one another. In music, this can be applied to pitch, rhythm, keys, or just about anything. For my initial research, I used a simple well-known one-line melody to analyze, *Mortal Kombat*. One can ignore the relatively simple rhythm and focus solely on the changing pitches for the first analysis, also ignoring octaves. See figures 1 and 2 for staff and text notation of this song.

Markov models can be represented using a *probability transition matrix*, in which the rates of occurrence of events are tabulated then normalized. Counting all of the notes that follows a pitch of A, one finds that A follows 16 times, C follows 1 time, D follows 1 time, E follows one time, F follows one time, and G follows 3 times. Normalizing this, one finds that the sum of this row is 23 and divides each term by 23. The final pitch-based Markov model of *Mortal Kombat* can be seen in figure 3.

This model provides a simple event probability for the input composition, but it lacks on any element beyond single pitches. Multiple Markov models could be used to model each of the musical elements, but maintaining many parallel models can become highly cumbersome. The largest problem with Markov models is that they address the how of musical composition, not the why. No computer model can address the why as of yet, however.

4 Lindenmayer Systems

Lindenmayer systems, or L-systems, operate on a grammar-based principal of music generation. First used in the late 1960's as a model for cell growth and plant topology, their pattern-based structure is being applied to new fields as well.

A simple case for a Lindenmayer grammar is provided as a set of an axiom and several productions, as in the example depicted by figure 4. Then, one runs iterations of this system, producing the output shown in figure 5. Concatenating all of these individual strings together, one can obtain a composition created by the system consisting of a string of notes. This is viewable as a text notation in figure 6, and a staff notation in figure 7.

The end result is highly mechanical and the inherent patterns are easy to spot. A quick way to circumvent this issue is to integrate a Markov model into a Lindenmayer grammar. By using the Markov model obtained from *Mortal Kombat*, one would begin to obtain the Lindenmayer grammar depicted in figure 8.

One could then randomly select one of the options based on the corresponding weights. The result composition would be similar in style (in this case, pitch-wise) to the original input composition, but there would be some variability in the output.

For this system to really produce complex music, it must also incorporate rhythm and harmony. These can be implemented using additional recursive Lindenmayer systems and Markov models, though other approaches are available as well. An eventual goal of this project is to successfully incorporate several important elements of music into the compositional system.

5 Genetic Algorithms

Using Lindenmayer systems with grammars produced from a Markov model will create music that mimicks the original. What does one do if s/he wants entirely original music?

Genetic Algorithms can be applied to evolve progressively sophisticated and pleasing music. Several compositions can be produced based on one input composition. A human user can then select the best productions and feed these back into the system, producing new music based on these. Eventually, as the system progressively generates music, the compositions should become more varied and more unique, resembling less and less the original input.

Such a prospect is likely far off, however, as there are still numerous hurdles to be jumped in effectively modeling music accurately.

References

- [1] McCormack, Jon. "Grammar Based Music Composition." Online. January 23, 2003. <http://www.csse.monash.edu.au/~jonmc/resources/L-systemsMusic.pdf>

Mortal Kombat



Figure 1: Input composition

A A C A D A E D C C E C G C E C G G B G C G D C F F A F C F C B
A A A A G C A A A A G E A A A A G C A A A A A A A

Figure 2: Text notation of input composition

	A	B	C	D	E	F	G
A	16/23		1/23	1/23	1/23	1/23	3/23
B	1/2						1/2
C	1/4	1/12	1/12		1/6	1/6	1/4
D	1/4		1/2	1/4			
E	1/4		1/2			1/4	
F	1/4		1/2			1/4	
G		1/8	1/2	1/8	1/8		1/8

Figure 3: Transition probability matrix

$a : C$
 $p_1 : A \rightarrow D$
 $p_2 : D \rightarrow D A E C$
 $p_3 : E \rightarrow E A A$
 $p_4 : C \rightarrow D$

Figure 4: Simple Lindenmayer grammar

Iteration	String
0	C
1	D
2	D A E C
3	D A E C D E A A D
4	D A E C D E A A D D A E C E A A D D D A E C

Figure 5: Running the Lindenmayer system

C D D A E C D A E C D E A A D D A E C D E A A D D A E C E A A D D D A E C

Figure 6: Text notation of Lindenmayer system composition

Lindenmayer Systems

♩ = 180

Figure 7: Lindenmayer system composition

$a :$		<i>any note in Markov model</i>	
$p_1 :$	A	$\frac{16}{23}$ $\frac{1}{23}$ $\frac{1}{23}$ $\frac{1}{23}$ $\frac{1}{23}$ $\frac{1}{23}$ $\frac{3}{23}$	\longrightarrow A \longrightarrow C \longrightarrow D \longrightarrow E \longrightarrow F \longrightarrow G
$p_2 :$	B	$\frac{1}{2}$ $\frac{1}{2}$ \vdots	\longrightarrow A \longrightarrow G \vdots
$p_7 :$	G	$\frac{1}{8}$ $\frac{1}{2}$ $\frac{1}{8}$ $\frac{1}{8}$ $\frac{1}{8}$	\longrightarrow B \longrightarrow C \longrightarrow D \longrightarrow E \longrightarrow G

Figure 8: Lindenmayer grammar using Markov model