Using Generative Grammar to Generate Music

A Study Of Stochastic Lindenmayer Systems

Sagnik Chakraborty
Indian Institute of Technology, Bombay

May 26, 2021

Abstract

Generative Music is a term introduced by ambient music composer Brian Eno[5], to describe music that is ever-different and changing, and that is created by a system. In this paper, we use Stochastic Context Sensitive Lindenmayer Systems to generate nearly a minute long MIDI music file, that is different with each run of the program.

1 Introduction

Panini's work on the grammatical structure of Sanskrit language in around 6th to 4th century BC can be regarded as the first attempt to give a formal, structural definition to grammar. According to philosopher and linguist Noam Chomsky[4], "even Panini's grammar can be interpreted as a fragment of a generative grammar."

However modern work on generative grammar began only in the 1950s. Chomsky is a pioneer in this field, and his book Syntactic Structures can be regarded as an introduction to the field of generative grammars and computational linguistics.[3]

Even though the study of generative grammar and formal languages was initially motivated by linguistics, around late 1950s-1960s it was found that

the syntax of programming languages can be described by one of Chomsky's grammar models called context-free grammar. Thus began the diversification of the applications of the theory of grammar from a purely linguistic standpoint.

It is now known that formal languages and grammar has applications in many fields, like molecular biology, symbolic dynamics and music[2]

In this paper we will show a program that takes the scale and beats per minute as input and produces around a minute long music clip in that scale. The music clip generated is different for every run of the program. Thus we produce generative music! We shall see how we used stochastic Lindenmayer (L) System, a type of formal grammar developed by theoretical biologist Aristid Lindenmayer[7] to model cellular division and fractal like growth of plants, to generate music as a language generated by generative grammar.

But first we aim to rigorously define some of the terms like "grammar" and "language" that we shall be using in this paper, and also differentiate between Chomskian grammar and L systems.

2 Formal Grammar

Define G as a 4-tuple (Σ, V, S, P) such that:
Σ is a finite, non-empty set of indivisible symbols called **terminal alphabet**, and the members of this set are called **terminals**
$V$is a finite, non-empty set disjoint from Σ. The elements of V are called **variables** or **non terminals**
$S \in V$ is a distinguished non terminal called the **start symbol**
P is a finite set of what we define as **productions** or **rules**, which are relationships of the form

$$\alpha \to \beta$$

where α is string of terminals and non terminals containing atleast one non terminal and β is a string of terminals and non terminals
Define P as **production set**
DEFINITION1 : A **sequential form** of G is any string over $\Sigma \cup V$
DEFINITION2 : Let γ_1 and γ_2 be two sequential forms over G. Then γ_1 **directly derives** γ_2 if $\gamma_1 = p_1 \alpha p_2$ and $\gamma_2 = p_1 \beta p_2$ (p_1 and p_2 can be null/empty) and $\alpha \to \beta \in P$. This is expressed as $\gamma_1 \Rightarrow \gamma_2$.
DEFINITION3 : Let γ_1 and γ_n be such that $\gamma_1 \Rightarrow \gamma_2 \Rightarrow ... \Rightarrow \gamma_{n-1} \Rightarrow \gamma_n$

where $\Rightarrow$ operator is to be understood as Definition 2. Then we say γ_1 **sequentially derives** γ_n. This is represented as $\gamma_1 \Rightarrow^* \gamma_n$. Also the sequence is termed as **derivation** of γ_n from γ_1

The **Language generated from Grammer G** , ie, $L(G)$ is defined as

$$L(G) = \{x | x \in \Sigma^*, S \Rightarrow^* x\}$$

where Σ^* is the set of all strings over Σ. The elements of L(G) are called *sentences* of $L(G)$. The context of usage of the term **Generative Grammar** for G is clear now, since possibly infinitely many sentences of $L(G)$ can be generated by the finite G .

Consider one such derivation from an example grammar G. For example, let

$$G = (\Sigma, V, S, P)$$

where,

$$\Sigma = \{0, 1\}, V = \{S, Q, L\}, S = S$$

and P is a set of the following rules:

$$\{S \rightarrow QLQ,$$

$$Q \rightarrow 1,$$

$$L \rightarrow 0,$$

$$1L \rightarrow Q11\}$$

$L(G)$ in this case, is finite and consists of *101* and *1111*. Observe that we carry out the derivations in a *sequential* manner as in

$$\underline{S} \Rightarrow \underline{Q}LQ \Rightarrow 1L\underline{Q} \Rightarrow \underline{1L}1 \Rightarrow \underline{Q}111 \Rightarrow 1111$$

Also note that two derivations may lead to the same sentence.
There are 4 types of grammar
Type 0: no constraints on G
Type 1: or *context-sensitive* grammar is a type of grammar G such that for every $\alpha \rightarrow \beta \in P, |\alpha| \leq |\beta|$ where $|s|$ denotes length of string s
Type 2: or *context free* grammar is a type of grammar G such that for every $\alpha \rightarrow \beta \in P, |\alpha| = 1$

Type 3: or *right linear or regular* grammar is a type of grammar, where every production $\in P$ has either of the three forms:

$$A \to cB, A \to c, A \to \epsilon$$

where ϵ denotes null/empty character, A, B are non terminals, and may or maynot be equal, and c is a terminal.
Note that the above example is a *context-sensitive* grammar, and if the last element of P is removed then it becomes a *context free* grammar.

3 L System

We have seen how in the Chomsky's model of formal grammar, sentences are **sequentially derived** from start symbol. Now consider instead the case where sentences are *parallelly derived*
As in

$$S \Rightarrow ((Q)(L)(Q)) \Rightarrow (1)(0)(1) = 101$$

At any instant of derivation, production rules are carried out parallelly throughout the entire length of the string. Note that for this particular example, the only sentence generated from G is *101* and the last production rule in P is irrelevant.
L-System[11] is a way to formally characterize this kind of parallel generative grammar.
DEFINITION4 : An *L system,* G is a 3-tuple

$$(V, \omega, P)$$

where V is a set of symbols containing both replaceable *variables* and non replaceable *terminals*. V is called the **alphabet**. Note L-systems do not distinguish between terminals and variables.
ω is a string of symbols from V defining the initial state of the system. ω is called **start, axiom or initiator**
P is the set of production rules to be applied parallelly during each step of the derivation.

We shall now distinguish between two types of L Systems. A **deterministic**

L system is one where after n derivation steps (denote by $\Rightarrow_n^*$) , initiator ω leads to an unique sentence $\lambda \in V$. A **stochastic** L system is one where

$$\omega \Rightarrow_n^* \lambda_1 or \lambda_2 or ... or \lambda_n$$

and there is a probability distribution π associated with each outcome λ_i.
Note that for a context-free L system, it is deterministic *iff* $\exists$ unique q for the production rule $v \rightarrow q$; $v,q \in V$.
For a context-sensitive L system, the above condition *does not guarantee it to be deterministic*, though context-sensitive L system grammars can always be made that satisfy the above condition, and which are deterministic.

4 Music Theory and Schenkerian Analysis

German music theorist Heinrich Schenker developed a theory of analysing tonal music which, with the benefit of hindsight and the development of formal grammar happening around three decades later, appears to be closely related to the theory of formal grammars[10][9].
The goal of Schenkerian Analysis is to demonstrate the organic coherence of the work by showing how it relates to an abstracted deep structure, called the Ursatz. This primal structure is roughly the same for any tonal work, but a Schenkerian analysis shows how, in an individual case, that structure develops into a unique work at the "foreground", the level of the score itself.
A key theoretical concept is "tonal space". The intervals between the notes of the tonic triad in the background form a tonal space that is filled with passing and neighbour tones, producing new triads and new tonal spaces that are open for further elaborations until the "surface" of the work (the score) is reached.
Let's start by drawing primary analogies. If we hypothesize that there exists a generative grammar G_M for generating tonal music, then Ursatz must be the *initiator*, ω. The notes are the *terminals* and the intervals between the notes, expressed as a generated string, should contain *variables* from which further music can be recursively derived in the next derivations. We can theorise that the grammar G_M generates music as:

$$\omega \Rightarrow^* background \Rightarrow^* middleground \Rightarrow^* foreground$$

If G_M is an L-system, it should be a desirable property that ω is infinitely

derivable, i.e., for any finite $n \exists \lambda_n$ such that $\omega \Rightarrow_n^* \lambda_n$ and λ_n contains both *terminals* and *variables*. This allows us to generate music of any arbitrary length from an initiator
In that case, foreground can be considered to be the string generated after sufficiently large derivations n and only the *terminals* denote the notes to be actually played, while the *variables* in the tonal space encode information about the duration of notes and other relevant structures.

The term *"the flowerings of diminution"* is used to describe the phenomenon in a book on Schenkerian Analysis.[6]

5 Generating Music

We have so far developed a theory that might help us in generating music, so now is the time to actually implement it in practice.
Firstly, some considerations:
1. L-Systems generate structures parallelly, and graphical interpretation of these structures (for example by interpreting the characters of the string as Turtle commands[1]) lead to the creation of fractal trees and space filling curves. But music is heard sequentially in time, so there must be an interpretation that allows us to *push and pop* states while rendering the music, rather than playing a cluster of notes together.
2. It is from practical experience that "good" music follows a sort of continuity of notes, as in the presently playing note is related to the immediate previous notes. Hence a *context-sensitive* grammar is more suited than a *context-free* grammar for generating music.
3. We would want to have a different music everytime we generate one. Hence a *Stochastic L-System* with *large number of possible generated strings for a large enough n* is best suited for our purpose.
Taking these considerations into mind, we first studied the context-sensitive L-systems described by Lindenmayer himself in his book *The Algorithmic Beauty of Plants*[1] to generate fractal trees. There are five examples of **deterministic** context-sensitive L-systems, given in his book in page 35.
All the 5 L-systems have the same $V = \{F, 0, 1, +, -, [,]\}$ and $\omega = F1F1F1$ but different *Production Sets, P*. An example of one such P is:

$$0 < 0 > 0 \rightarrow 0$$

$$0 < 0 > 1 \rightarrow 1[+F1F1]$$
$$0 < 1 > 0 \rightarrow 1$$
$$0 < 1 > 1 \rightarrow 1$$
$$1 < 0 > 0 \rightarrow 0$$
$$1 < 0 > 1 \rightarrow 1F1$$
$$1 < 1 > 0 \rightarrow 0$$
$$1 < 1 > 1 \rightarrow 0$$
$$+ \rightarrow -$$
$$- \rightarrow +$$

where $<, >$ do not denote any character but rather the instruction to look for the preceding(in case of $<$) or succeeding(in case of $>$) 0 or 1 while ignoring the other characters *(F,+,-,[,])*.
But we need to make the grammar **stochastic**, so what we do is to define G_M as

$$(V, \omega = F1F1F1, P_1, P_2, P_3, P_4, P_5, \pi)$$

where P_i is one of the five production rules and $\pi : \mathbb{N} \rightarrow \mathbb{P}$ is a probability function that maps n denoting n^{th} derivation to the set $\mathbb{P}$ where,

$$\mathbb{P} = \{P_i | 1 \leq i \leq 5\}$$

So for each n^{th} derivation, any one of the five production rules are used to generate the next string.

Using this generative grammar, for a depth of 6 derivations we get the following string for one arbitrary run of the program

$$F1F0F0F1[-F0F1[+F1F1]]F1$$

We interpret the string musically as follows:
The string is to be read from left to right
F denotes the instruction to increase tone duration by a quarter note.
1 and 0 are to ignored while rendering the music
+ denotes the instruction to raise the current note by one step in the specified key, and - is the instruction to lower it

[is the instruction to push the current state(present note and present duration of the said note) into stack and set duration to zero
] is the instruction to pop from the stack and ***play*** *the popped note for the popped duration*

Using this interpretation, we were able to generate music that both sounded sufficiently musical as well as had the basic structure of dissonance, resolution and suspense, thanks to the Schenkerian interpretation of the generating grammar that we used. The python code created for the project as well as some sample music generated by this program, can be found at https://github.com/sagnikiitb/generative$_music$

Some Notes on π Function:
$Initiallyweassignedaproductionrule\mathrm{P}_i$ for each derivation, and for each derivation step, each P_i was equally likely to be assigned. However this led to the music sounding a bit chaotic. So we split up into intervals of length k and for each interva, a P_i was uniformly assigned, and that production rule was followed throughout the interval. For example if $k = 5$, then at the production rule P_2 (say) is followed for the first 5 derivations, then the production rule P_1 (say) is followed for the next 5 derivations, and so on. By experimenting with different values, $k = 5$ was subjectively found to result in the most melodic sounding music, while leaving sufficient room for randomization.

6 Acknowledgement

We would love to acknowledge the authors of the paper, *"Growing Music: musical interpretations of L-Systems"*[8], for providing the motivation and direction to do this project, through their paper.

References

[1] P. P. Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. 1990.

[2] M. Baroni and Drabkin. *The Concept of Musical Grammar Music Analysis*. 1983.

[3] N. Chomsky. *Syntactic Structures*. 1957.

[4] N. Chomsky. *Aspects of the theory of syntax*. 2015.

[5] B. Eno. 1996.

[6] O. Jonas. *Introduction to the Theory of Heinrich Schenker.* 1972.

[7] A. Lindenmayer. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. 1968.

[8] S. S. Peter Worth. Growing music: musical interpretations of l-systems.

[9] H. Schenker. *Elucidations(Erläuterungen).*

[10] H. Schenker. *Free Composition (Der Freie Satz).*

[11] A. Walker. Adult languages of l systems and the chomsky hierarchy.