Theorizer – A tool that abstracts fundamental concepts in music theory and can be used in a procedural music composition environment.

Ramakrishnan Subramanian

Advisor: Luke Dubois

In partial fulfillment of the requirements for the degree of Master of Professional Studies at the Interactive Telecommunications Program at New York University

May, 2006
Abstract

The proposition I intend to explore in my thesis is that 'a rule-based intelligent abstraction of fundamental concepts of music theory can be a powerful tool for sound artists, computer music composers and programmers alike.' In order to investigate this, I will build an open-source interactive software application that will abstract certain concepts in western classical music and Indian carnatic music theory. My primary motive is to ensure interactivity through intelligent response and an intuitive interface while maintaining originality in the rendering of musical sequences.

Personal Statement

The average middle-class Indian does not listen to western music. The average middle class Indian cannot name five western musical instruments. Not many pop/rock artists tour India and the amount of exposure to the western culture is minimal.

In the late 90's and early 2000's, the local music scene consisted of a lot of cover bands and very few bands playing original music. Exposure to experimental music, even electronic music was minimal. Cover bands were all over the place (the band that yours truly played for included ;) and at every concert the performers would get slippered or fall out with the organisers if their set did not include a significant filling of covers. There was and still is a distinct lack of originality in the stuff that western and fusion bands came up with that stuck with me.

Inspired by Eric Clapton's unplugged version of 'Layla' that i watched on MTV for the first time in 1997, I picked up a guitar at age 14. I took private lessons from a renowned Gospel artist for a year. After that I basically taught myself, spending hours each day with my instrument composing different tunes. I grew and matured as a musician playing and recording with various artists and performing at different venues all over India. At 18, I started teaching at one of Chennai's most reputed music schools 'The Unwind School of Music'.

I realized all along that I was not as gifted a musician as some of my seniors and contemporaries I was playing and performing with, but my knowledge of musical theory was far superior to the others. It proved to be one of my strengths. I acquired better musical taste as i went along, but my compositions always stemmed from the fact that i knew what would sound good/correct/in harmony just based on
compositional rules. By trying out various options, I eventually arrived at what I thought sounded best. These decisions were influenced by my listening skills as well and the roots became stronger with experience. I also started associating intervals with emotions. I always associated mapping techniques to remember, play or create a musical sequence.

During my numerous recording experiences, I came across various signal processing and effects processing equipment for voice and musical instruments. After a little research into the working of these devices, it occurred to me that I could transcend boundaries and incorporate my learning on the technical front into creating advanced art forms. Delving into the subject further, I learnt about some aspects of Computer Music Synthesis as to how it can be used for one's own ideas, freedom of expression, precision and greatly enhanced implementation of new ideas. All my ideas at this point of time were only centered on sound synthesis and creating interesting effects and listening experiences like simulating a musical instrument and the like. This was in 2003.

I went to engineering school in India and my approach to almost everything borders along an engineer's approach—logical reasoning, analytics, efficiency and usability. But it was also characterized by a narrow-minded approach to any endeavor. Seeking to break out of this, and broaden my creative horizons I went to grad school at Interactive Telecommunications Program. I did a fair amount of work with sound and MIDI and intelligent listeners but I was first exposed to real time composition with audio synthesis only a year into Masters Program, fall 2005. The whole idea about being able to write a computer program simulates the experience of listening to a live guitar solo by Van-Halen fascinates me.

The growing availability of computer applications incorporating audio particularly over the Internet and in portable devices has increased the need for an ever-wider group of engineers and computer scientists to understand audio signal processing. One of the aspects of digital audio and video that fascinates me is the wide range of disciplines it embraces. There is still no clear-cut explanation of the mechanism that designers use to combine technologies to make a product, and an affordable one at that. Whilst the theory we use is scientifically based, the actual purpose that it serves can encompass varied disciplines.

Though all the knowledge I have of audio programming and audio
signal processing is from an application standpoint and not necessarily from a research perspective, I intend to use it to create a usable, user friendly and experience enriching applications.

**Context**

I would like to start expressing my ideas and research in this paper by first trying to answer a couple of questions that are very relevant to my work.

i) Why automation of music compositions?

ii) Why use computers to make music?

The reason why some notes sound pleasant together and why some combination of notes make us cringe can be attributed to mathematical ratios that represent the wave patterns created by the instrument. This is the most basic representation of mathematics in music. These ratios have helped determine tuning systems, explain intonation and other musical deterministic properties.

There have been attempts to sonify various mathematical equations ranging from permutations and combinations and PASCAL’s triangle equations to markov equations and various stochastic processes using a simple mapping technique that uses a normalized range of numbers mapped to musical notes using a human determined mapping function.

The possibility of composition using techniques like iteration and recursion minimizes the amount of human intervention. Automation of music composition has been a fascination of man for a long time. The advent of computer technology has made the algorithmic composition process in music more sophisticated using complex calculations. These complex calculations have been based on pre compositional resources: stochastic processes, aleatoric methods, genetic algorithms, fuzzy logic, pattern matching, spectral modeling and artificial intelligence.

Greeks invented the first automatic composition machine with the Greek Aeolian harp. This may be considered the first automatic instrument. It was named for Aeolus, the Greek god of the wind. The instrument had two bridges over which the strings passed. The instrument was placed in a window where air current would pass, and
the strings were activated by the wind current. Rather than being of different lengths, the strings were all the same length and tuned to the same pitch, but because of different string thicknesses, varying pitches could be produced. This was in 500 BC.

Since then regular innovations have followed in the field of automation of composition. The earlier innovations were physical instruments that used various abstraction-> alphabets mapped to the sound frequencies and other similar mnemonics. Pythagoras discovered numerical ratios corresponding to intervals of the musical scale. He associated these ratios with what he called "harmony of the spheres."

In the 1400's Isorhythmic motets were developed. Isorhythms are a form of music in which a given rhythm cyclically repeats, although corresponding melody notes may change. These songs made use of patterns of rhythms and pitches to define the composition. Composers like Machaut (14th century), Dufay and Dunstable, (15th century) composed isorhythmic motets. Duration and melody patterns, the talea and the color respectively, were not of identical length. Music was developed by the different permutations of pitch and rhythmic values. So if there were 5 durations and 7 pitches, the pitches were lined up with the durations. Whatever pitches were 'leftover,' got moved to the first duration values. The composer would permute through all pitches and durations before the original pattern would
Synthesizer technology advanced with new inventions and complex DSP chips to better synthesize and create new sound forms and waves. The computer induced a radical change in electronic and computer music. Computer programmers could now sequence sounds since the computer itself is powerful enough to synthesize sounds. It dawned upon computer music artists that they could now either sequence sounds in a computer which is a procedural form of composition where the entire composition is scored by the user or use artificial intelligence to sequence those same sounds. The latter has popularly been referred to as algorithmic composition.

Bruce L Jacob (1996) writes an interesting definition for algorithmic composition

_Algorithmic composition is the application of a rigid, well-defined algorithm to the process of composing music. It is frowned upon by `traditional' composers because it is often used as a means to expand one's musical palette. The explicit message is that algorithmic composition is invalid as a methodology, but the implicit message is that the music produced is in some sense unlike what the composer would have produced without help. In other words, music produced by algorithmic composition is considered somehow inferior not because it was produced by an algorithm, but because it is someone else's music--it belongs to the designer of the algorithm, and not to the user of the algorithm. One can avoid this criticism merely by implementing one's own algorithm."

There are popularly two different methodologies that exist in computer-generated algorithmic composition: (1.) "stochastic" vs. (2.) "rule-based" systems. There is also a third category, (3.) which
we can label AI, or artificial intelligence systems.

This idea of building small, well-defined compositional functions—i.e. "subroutines"—and assembling them together would prove efficient and allow the system a degree of flexibility and generality, which has made this approach a popular technique in algorithmic composition systems.

There is also another facet to computer music synthesis and algorithmic composition. It can be referred to as the ‘listener’.

"Another tendency is to use the computer as an accompanist who listens to what is being played and responds appropriately in real-time. Here, the human input is used to generate rules on which the machine will base its output. This is seen in such programs as Cypher (Rowe, 1993) and IBLSmart (Widmer, 1994)" (Jacob, 1996).

My early attempts at implementing my own algorithm to compose music lacked any musical structure and definition. Using computers to compose music is a tremendous experience and it really broadens one’s musical creativity in terms of theoretical knowledge and expression. My rather futile attempts involved composing using rule-based systems. Working incessantly, tweaking the rules to eventually obtain the desired output, it was almost like was scoring the composition in full and not really having the algorithm compose the musical sequence. A clear understanding of how rule based systems and expert systems worked was lacking.

Modeling style to simulate certain features of a human player is a challenge in algorithmic composition. This can be achieved by introducing artificial intelligence into the playing. The most common method is to analyze scores of existing music whose style the program is trying to replicate and extract features for reference. Also, through some form of feedback the program is able to determine whether it made the right choices. This will influence future decision making of the program.

Kristine H. Burns (music composer and author) has this interesting comparison between rule-based systems and artificially intelligent systems from a composer's standpoint.
"The study of Artificial Intelligence in computer music warrants a comparison between Rule-based algorithms and Artificial Intelligence (AI) algorithms. Although rule-based systems make use of a series of steps to reach the logical conclusion, and may even involve back-tracking schemes in which the program is able to "back up" and begin a portion of or all of the steps again, this in no way implies that a rule-based program is learning. Computer science has "sought to find ways to embody knowledge in machines. . ." Knowledge-based, or AI, systems are those in which the computer algorithms are able to "learn" which solutions are retainable/usable by a series of comparisons with previously-stated material. This type of programming is often referred to as an "expert system" because "they're [expert systems] based on imitating the methods of particular human practitioners."

As with any art, algorithmic composition involves practice and self understanding. Is it possible to approach learning algorithmic composition in the same manner as any other instrument? The question is... instead of playing a few chords or a melody line, can a few lines of code and logic can evoke a similar expression in the listener?

A plethora of artists have done it before and done it with wide variety. The interesting thing about using computers to compose musical sequences is that it is possible to simulate certain pitch variations and rhythm sequences that are not physically possible for the player. Speed apart, traversing between three different octaves on successive notes is difficult for a guitar player. Likewise, splitting a unit of time into five equal intervals is very difficult for a human composer.

What I think I have not seen enough of is transcending boundaries to define rules for each instrument. The way the guitar-player approaches a jazz composition can be different from how a bass player approaches the same composition. Rules for a rock guitar player can be very different from rules for a jazz-guitar player. What happens when you combine rules defined for a rock-guitar player to rules defined for 'Carnatic music theory'? Rock-guitar rules might define dominant intervals used by rock guitar players, chord structures, chord transitions, popular triads and rhythm notations. Carnatic music has rules that define ascending and descending patterns in scales. Combining results of both can be very interesting
According to David Cope, a style consistent algorithmic composition is composed of the following three tasks.

1. Deconstruction (analyze and separate into parts)
2. Signatures (commonality - retain that which signifies style)
3. Compatibility (recombinancy - recombine into new works)

Cope has created most of his work using the above concepts and they have been the benchmark for many an aspiring computer music composer.

I propose that using a rule based abstraction of the results of musical analysis using a procedure that is derived from Cope's theories offers the flexibility to combine different styles minimizing dependency between the different processes and the results of the different processes.

An analysis of a particular score can reveal results about the following

i) pitch
ii) rhythm
iii) phrasing techniques

Information about these three properties is sufficient to adequately define a particular style.

An intelligent abstraction of the results of analysis of various compositions can be laid out in formats that can be combined with results from other analysis. This can lead to novel music expressions.

The methodology section of my paper will outline clearly my technical approach to solving this problem.

**Methodology**

Consider the function below

\[ X_n = f(x_{n-1}, x_{n-2}, x_{n-3}). \]

(Where every note that the algorithm out is a function of the last three notes that it has played.)

The function can be any mapping function that is determined by the user. For instance if all the notes in the music scale are assigned
some sort of weight to determine level of dissonance/consonance, then the function could be something as simple as the addition of weights for 4 successive notes adding up to a multiple of 0.5. (The sum of the weights of all the notes with respect to the tonic add up to 1)

*Composing by phrases:*

It might be more interesting if we use this to create a pattern in every rhythm unit (tweaking it at the end of a rhythm unit) rather than apply it for an extended duration.

In sheet music terms, the rhythm unit is referred to in ‘bars’ which are just one reference of time measure that is dependent on the tempo.

The program must intelligently be able to identify the termination of a sequence based on its position in the time domain. Rules of consonance and dissonance, if followed, will make sure that the sequence conforms to basic rules of tension and relaxation. Certain intervals cause tension in the listener and this tension is relaxed when there is appropriate resolution with respect to the tonic.

Composing by phrases makes it easier to implement a certain style and expression. Therefore we can clearly make a distinction between successive notes that occur in the same phrase and not apply the above function to notes which are spread out across two different phrases.

*Scale Matrix*

I propose three layers of abstraction. On a basic level we analyze the music of a particular genre and come up with a Markov set that provides a weighted probability of a transition from one note to the other. Therefore we would have a 12*12 matrix with each having a particular probability so we instantly have a set of 144 probabilities. So during the composition of any phrase we can use this set to come up with the genre style-consistent next note. This would be distinct for each type of scale. A major scale would have a probability matrix; a minor scale would have a probability matrix and so on.

For e.g., this would be the sample scale matrix for a song composed in C major. Each note has a probability that represents the probability
A layer above this level of abstraction would be to identify patterns played by specific instrumentalists. Why? Because the method by which a guitar player approaches classical music is different from a piano player doing the same. Therefore the key is to identify playing patterns of a guitar player to distinguish him/her from a piano player within each phrase. This provided added intelligence to the system. By analyzing the patterns within each bar we will obtain another 12 * 12 matrix of probabilities. But instead of mapping note against note and finding out the probabilities of transition from among a complete set, it would make more sense if we could map intervals.

The interval set could be just broken down into seven brackets.

- (Minor/major 2nd) -> 2
- (Minor/major 3rd) -> 3
- (Minor/major 6th) -> 6
- (Minor/major 7th) -> 7
- (Perfect/ague 4th) -> 4
- (Dim/perfect/ague 5th) -> 5

The same is obtained for descending intervals.

We would still have a 12*12 matrix but from this matrix we get an
idea of the player’s common interval patterns. From this matrix one can obtain the probabilities of all the intervals from any given note with respect to a particular tonic.

This interval abstraction can be combined with a scale of given choice and the corresponding note to interval mapping can realize the exact musical notes.

Sample composition flow:

Composing a classical violin sequence would entail the following steps.

1. Set all initial parameters (Key, tempo, duration, style)
2. Load instrument sounds for a violin.
3. Identify signatures from violin analysis results and create a score.
4. Use the DSP functionality to play each note in the score

Consider a sequence extending for four bars and starts with the tonic.

The most probable interval is obtained from the interval matrix and the corresponding note is obtained from the scale set originally.

If the note interval obtained from the Interval Matrix does not correspond to a note on the scale matrix, the closest note will be picked until a tolerance limit is reached. Once that is broken then accidentals are introduced to maintain consistency on both fronts.
For instance, oriental/Chinese music is composed exclusively on the pentatonic scale. So if we were to consider a jazz player's interval matrix and an Oriental Scale matrix the results would be a modified pentatonic scale almost making it bluesy. This is the flexibility that this form of abstraction provides.

Rhythm layer

The idea of associating rhythms is interesting if entire patterns are analyzed. Rhythm patterns within a particular phrase can be identified and then fed into a simple neural network.

Each pattern can be marked up
- \( qqqq \rightarrow 4\) quarters
- \( qqrrqqr \rightarrow \) quarter, quarter rest, quarter, quarter rest and so on.
  *This is specific to a player's style.*
- \( qeeqqq \rightarrow \) quarter, eighth, eighth, quarter, quarter, quarter, quarter.

The algorithm is intelligent enough to evaluate the time left in each bar and pick out a pattern accordingly. Therefore the rhythm pattern generated would be a function of the duration values generated and the amount of time remaining in the phrase, while at the same time being consistent with the patterns generated. For the sake of the methodology this will be a simple neural network minimizing the layers of complexity.

Idiosyncratic layer

The third layer of abstraction would be specific idiosyncrasies of the instrument and the player. For example analysis of specific pitch-varying techniques of a player and the frequency of use of those techniques will certainly lead to more style consistent patterns.

For example to reproduce a guitar-bend intelligently in a phrase we would have to analyze the duration of the bend and the number of intervals it encompasses. These sort of pitch-varying expressions are specific to instruments. String bending expressions are specific to

The most important sections in creating this idiosyncratic layer
- *i)*  Model amplitude varying techniques
- *ii)* Model pitches varying techniques.
With these three layers of abstractions we could come up with an interesting expression by just combining the rule-sets. For instance, if we wanted to evolve Van Halen playing Carnatic music, then we would feed in Van Halen's playing patterns to a basic carnatic theory rule set (like a rule for a scale) and we would obtain a composition that employs rock guitar playing style following carnatic scales and melody rules.

The union rule-set might have some conflicts but it may yet provide for a plethora of new expressions.

This layer based abstraction of music theory compositional rules and styles also make it easy for the programmer/composer to

1) combine rule sets to come up with a unique style
2) Modify existing rule sets (probability matrices and transfer functions to define a new style)
3) Model a new style based on existing styles.

By this means it starts to become easier to compose new music as it is equivalent to putting blocks into place.

Having studied the works of different composers-programmers like David Cope, Karl Heinz Essl, Brad Garton, John Cage and Paul Lansky i am convinced there is scope for algorithmic composition tools to be used in a live composition and performance environment simulating different instruments and different styles and at each instance being faithful to the style of the composer.

**Technical details**

The process consisted of three steps

- Score analysis
- Instrument and style modeling
- Procedural composition

The score analysis process is as outlined. A sample score analysis input file looks like the one shown in the example below.
When a midi file is input to the analysis program, the following information is extracted out of every note that is played.

i) Pitch
ii) Amplitude
iii) Channel
iv) Pitch shifting information (bend)
v) Duration

A python script parses through this information to

- Obtain Probability Matrices (scale matrix, interval matrix)
- Identify patterns in a definite time period
- Determine signatures (commonality - retain that which signifies style)

With this information, a database of abstractions from the analysis of the score file is obtained. The above operations are performed for a series of score files to obtain a significant set of abstractions.
encompassing different styles and genres.

**Style Modeling**

- *Time Irregularities*
- *Modeling accentuations, amplitude variations*
- *Generate patterns based on existing patterns obtained from score analysis*

It is necessary to model the fundamental randomness of human nature. Imperfection is a salient feature in musical composition. Therefore to emulate that human feature it is important for instance, to stagger the entrance of each note in a chord played on a stringed instrument.

A neural network package Joone (Java open source Neural Network package) was used to generate similar patterns based on existing patterns provided as input. A simple back propagation neural network was designed and used.

*Software Overview*
Compositions:

I have three pieces composed with this software

i) Random-less
ii) Kar-naughty
iii) Eruption-II

Random-less is a piece composed using this piece of software with the following presets, the key: C# minor and the duration of each note were set to a constant value. It was inspired by existing algorithmic compositions that were inspired by random and stochastic processes and aleatoric music

Notes:
- Instrument sounds modeled based on the Karplus-Strong Algorithm
- Underlying pitch-set is coherent
- Amplitude curves are modeled to replicate a live instrument
- No definite correlation between successive pitches

Kar-naughty is a piece inspired by Indian classical music and the
complexity of rules that govern music in carnatic music. The instrument sounds were modeled using RTcmix’s instrument interface and this particular piece had two tracks- a melody track and a synth backing, both created using the theorizer software package.

Notes:
- Based on analysis of Indian carnatic scores
- Not diatonic, Melody lines dominate the entire score
- Complicated rule-set ascending pitches follow a certain set of rules, descending pitches follow another set of rules

Eruption-II is a piece designed to mock/model a rock guitar. For this piece I analyzed scores of Van Halen’s guitar solos and came up with sequences that could replicate that particular style. The entire piece is a call and response composition, the sections being played by Eddie and the ‘theorizer’ program.

Glossary:

I have provided a detailed explanation of some of the concepts that I have used extensively in my research and development process

Markov Chains:

A stochastic process with a finite number of states in which the probability of occurrence of a future state is conditional only upon the current state; past states are inconsequential. In musical composition, Markov chains have been used to describe a distribution in which the state at time step n + 1 is determined only by choice of note at state n.

A Markov chain is a sequence \( X_1, X_2, X_3, \ldots \) of random variables with the property (Markov property): the conditional probability distribution of the next future state \( X_{n+1} \) given the present and past states is a function of the present state \( X_n \) alone, i.e.:

\[
Pr(X_{n+1} = x | X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n) = Pr(X_{n+1} = x | X_n = x_n).
\]

The range of the variables, i.e., the set of their possible values, is called the state space, the value of \( X_n \) being the state of the process at time \( n \).

Back Propagation neural networks:
Back propagation is a **supervised learning** technique used for training **artificial neural networks**. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors". Backpropagation requires that the **transfer function** used by the **artificial neurons** (or "nodes") be **differentiable**.

The summary of the technique is as follows:

1. Present a training sample to the neural network.
2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a **scaling factor**, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat the steps above on the neurons at the previous level, using each one's "blame" as its error.

**Fourier series**

Fourier series is a way of analyzing a periodic function by decomposing it into a weighted sum of simpler sinusoidal component functions.

Many other Fourier-related transforms have since been defined, extending to other applications the initial idea of representing any periodic function as a superposition of harmonics. This general area of inquiry is now called harmonic analysis.

The weights, or coefficients, of the sinusoidal components, are a one-to-one mapping of the original function and hence varying these coefficients can lead to different sounds provided the frequency falls in the audible range. This process is widely referred as harmonic composition.

**Modulation**
Modulation is the process of varying a carrier signal in order to use that signal to convey information. The three key parameters of a sinusoid are its amplitude, its phase and its frequency, all of which can be modified in accordance with an information signal to obtain the modulated signal.

FM:
Frequency modulation (FM) is a form of modulation which represents information as variations in the instantaneous frequency of a carrier wave.

AM:
Amplitude modulation (AM) is a form of modulation in which the amplitude of a carrier wave is varied in direct proportion to that of a modulating signal.

References

Conclusion:
This research has brought forth interesting results till now and through experimentation and further research there is further scope for producing style consistent compositions of other genres like blues and jazz.

The initial goal was to try and create an abstraction of musical concepts such that it would make it easier for a musically illiterate person to indulge in computer music composition. However at this stage it seems like it would take as much time for a musically illiterate person to come up with a structured music composition using this tool as it would take them to be come musically literate by learning an instrument.

The amount of human intervention required to compose a musical composition in its entirety was/is much more than anticipated. I intend to further my research to reduce this mode of interaction to an extent.