Algorithmic Composition: Research and Reflections

David Chambers

Metsavälu
Põllküla
Keila Vald
Harjumaa
76712
Estonia
chambers.dave@rocketmail.com

Abstract: This article describes Computer Aided Algorithmic Composition (CAAC), outlines a brief history and recounts a concise taxonomy (put forward by Christopher Ariza). A CAAC system is principally defined by two elements – the *algorithm* and the *mapping*. The research of Jon McCormack is used to offer the reader some insight into the actual *operation* of an algorithm, considered in the context of a CAAC system. Markov chains, a stochastic process, L-systems, a string rewriting grammar and hierarchical grammars with two rule sets are exposed. Some recent (2007) research by Alexis Kirke and Eduardo Miranda is then examined in the critical area of *mapping*. The Game of Life (GL) provides an exemplar for Kirke and Miranda's discussion of the Cellular Automata system. The final section offers general reflections, outlines a possible *user-interface* and concludes with thoughts on what the future of CAAC systems might hold.

1 Introduction

The Oxford Dictionary asserts an algorithm is 'a process or set of rules used in calculations or other problem-solving operations'. Algorithmic Composition (AC) refers to both computer-aided (CAAC) and non-computer-aided, such as rolling a dice, composition. Although the use of a computer for music production is by no means mandatory, this paper is concerned with the formation of musical structures and the 'calculations' performed by the algorithm will operate on data that at some stage possesses musical meaning. A plethora of AC models exist including knowledge-based systems, fractals, systems that learn, even systems based on natural selection and those that harness genetic processes (two well-known examples of heuristic algorithms are genetic algorithms and neural networks - both of these are based on the simulation of what scientists understand from biology). CAAC systems are a subset of AC systems. The success or failure of these systems as sources of inspiration for composition largely depends on the *mapping* system employed.

Mapping is the process of aligning two distinct data sets - in this case musical and non-musical data. A CAAC system allows the user to manipulate musical representations. Indirect representations are mapped or transformed into direct representations such as MIDI (Musical Instrument Digital Interface) note data or digital audio files.

AC can be traced back to Ancient Greek formalisations and more recently Mozart used stochastic techniques in his Musikalisches Wurfelspiel, rolling a dice to determine notes. Darreg's 1957 article *The Electronic Computer Takes up Music Lessons* contemplated the future of CAAC systems remarking 'a computer can only do what it has been instructed to do--and explicitly instructed. But this does not lessen the value to composers, arrangers, and others of a new and highly useful tool.'[Da57] Cynics may disagree, inferring that to 'establish a system for representing any aspect of music is almost a challenge to a composer to invent music which the system is incapable of representing.' [Ma00] However, the author agrees with Jon McCormack when he claims it is realistic to 'use the machine as a synergetic partner to the human composer.'[Mj96]

2 Ariza's Taxonomy

The purpose of Ariza's[1] classification was certainly not to create mutually exclusive 'types' of CAAC system. His motivation seemed to be the opinion that previous attempts failed to isolate attributes from the perspective of the user and developer. He offers seven descriptors with which to identify systems but asserts that 'all CAAC systems have some aspect of each descriptor.' [Ar05] The descriptors are as follows:

(1) ***Scale***:
Does the CAAC generate Micro or Macro structures? Micro structures are musical event sequences commonly referred to as sound objects, gestures, textures, or phrases: small musical materials that need including in larger structures. Macro structures are musical event sequences that tend towards complete musical works.
(2) ***Process-time***:
Whether a system is Real-Time or Non-Real-Time. Ariza offers the useful distinction that reversing a string of events cannot be performed by a RT CAAC as this would require knowledge of the future.
(3) ***Idiom-affinity*:**
The extent to which the CAAC is targeted towards a certain musical idiom, style, genre or form.
(4) ***Extensibility*:**
Can code be added? Perhaps a subclass of a system defined object for an object-orientated subject. Systems may be described as open or closed. All open-source systems have open extensibility and closed systems may or may not provide this.

---

[1] These seven descriptors are now the 'System Filter' criteria on Ariza's excellent and comprehensive CAAC website, Algorithmic.net [2]Source: [Mc96]

(5) ***Event Production*:**
Generation and Transformation. A distinction is made between the generation of events from indirect music representations (possibly algorithms or lists) and the transformation of direct music representations (such as MIDI files).
(6) **Sound Source:**
Systems with onboard instrument algorithms are capable of internally producing sound. There is a distinction between sound production within a system and event production. Some systems may be capable of both. Some systems may export the definitions of a sound to an external system that actually generates the sound. Alternatively, an external system may produce the sound from it's own synthesis engine. To summarise, the possibilities are: Internal, Exported, Imported and External.
(7) **User Environment:**
Language, Batch and Interactive. A language interface provides the user with an artificial language to design and configure music abstractions. A batch interface fails to provide interaction and only permits the user to provide input data. An interactive interface allows the user to issue commands and, for each command get a response.

3 Algorithms

There are five types, or processes, of algorithms - stochastic, chaotic, grammars, artificial intelligence and rule-based. The work of McCormack [MJ96] is particularly interesting as it describes 'a computer assisted music composition system' (essentially a CAAC) that combines three of these five types – stochastic, grammar and rule-based algorithms.

3.1 Stochastic

Markov chains are a stochastic process and were conceived as a means by which decisions could be made based on probability. The probability of a specific event i is context dependent on the occurrence of a previous event. 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. Each element in the transition matrix, *Pij* 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.

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 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: Sequence of note events and corresponding transition matrix[2]

Counts in each row are normalised so the combined probabilities for each row sum to 1:

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

Crucially, the composition will have to be analysed first in order to calculate the probabilities unless we wish to fill the transition table with randomly generated probabilities.

3.2 Grammar and Rule-based

L-Systems are a form of string rewriting grammar. Letters are replaced one at a time sequentially across the string corresponding to the serial nature of the process that the grammar represents – for example the sequence of input notes in figure 1.

McCormick adapts *parallel* rewriting grammars for composition where rules are applied in parallel to all letters. The process begins with a seed. Figure 2 shows that after the 2nd iteration, each of C, G and C is operated on simultaneously to yield two E's (G tends to the empty word).

This iterative process continues for a given number of levels.

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

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

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

Figure 2: Rules of the grammar, iteration table and resultant concatenated string

So far the process has been deterministic. McCormick's system also employed stochastic grammars that map a set of probabilities. Figure 3 shows for instance that with production p1, C may be mapped to E only one in three times. With this implementation the algorithm becomes stochastic and a Markov model can be applied.

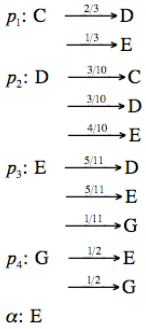


Figure 3: Stochastic grammar equilvalent to the transition matrix in figure 1

Hierarchical grammars occur when a hierarchical set of rules is imposed. The motivation for doing this is to more accurately represent the different *levels* and the complex patterns prevalent within a musical composition. McCormick applied hierarchical grammars in the development of his CAAC system. Schematics for which are shown in figure 4 below.

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)
}

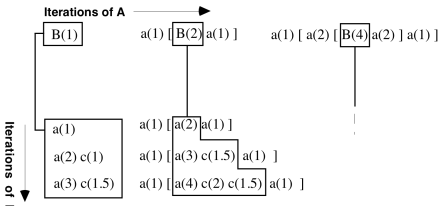


Figure 4: Hierarchical grammar rule sets (left) and resultant strings (right)

3.3 Discussion

Whilst the inclusion of McCormack's work was to illustrate the mechanics of some algorithms, it is worth saying a few words about his findings. In his own words, the 'parallel rewriting technique of L–grammars has capabilities equal to, or better than, those of previous grammar based systems.'[Mc96] McCormack's belief that a hierarchical system was better served to emulate the different levels (micro to macro) within a composition was affirmed by the research of Farbood and Sconer who claim Markov chains are well suited for generating counterpoint, adaptable and 'provide the flexibility of integrating deterministic and probabilistic rules in the same model' [FS01]. The analysis phase of the system described is not always present and the rules of a an CAAC systems could be pre-coded into the software or edited in real time by the end user, offering control over the probability of certain events.

Glitches in the audio stream may be detected when very complex grammars are parsed but McCormack suggest saving data to disk for later playback. Increasingly powerful computers should help alleviate this issue and realise the goal of real-time performance.

4 Mapping

The Game of Life (GL) is a two-dimensional automaton invented by John Conway. Conway applied simple rules (shown beneath figure 5) to the cells of an [m x n] matrix and produced patterns that would 'expand, change shape, or die out unpredictably. He wanted to find the simplest possible set of rules that would give such an interesting behaviour.'[KM07] Black cells are living and have a binary representation of 1, white cells are dead and represented by 0.

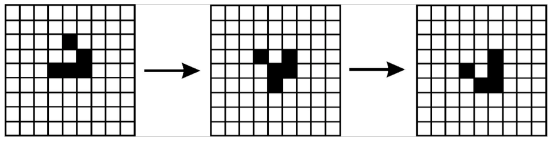


Figure 5: Game of Life is a two-dimensional cellular automaton where each cell can be in one of two possible states: alive or dead

The state of a cell as time progresses is determined by the state of its eight nearest neighbouring cells, as follows:
· Birth: A cell that is dead at time t becomes alive at time $t + 1$ if exactly three of its neighbours are alive at time t
· Death by overcrowding: A cell that is alive at time t will die at time $t + 1$ if four or more of its neighbours are alive at time t
· Death by exposure: A cell that is alive at time t will die at time $t + 1$ if it has one or no live neighbours at time t
· Survival: A cell that is alive at time t will remain alive at time $t + 1$ only if it has either two or three live neighbours at time t

4.1 Methods

All CAAC systems deal with *event* production as defined by descriptor 5 of Ariza's Taxonomy. Kirke & Miranda (2007) used GL to generate events and illustrate the importance of the mapping method that converts these events to musically meaningful data. They argued that the mapping could 'be adjusted in such a way as to vastly change the properties of any generative compositional system.'[KM07] GL data is normally mapped by binary or Cartesian methods. Kirke & Miranda introduced a new 'radial' mapping (see figure 6) based on a 'desire to capture the aesthetic of GL in a non-visual art form' [KM07].

The new mapping used a complex number system with the origin placed at the centre of the GL board. Each live cell generated data based on its (r,θ) co-ordinate rather than its (x,y) co-ordinate. The rationale behind the research into the new mapping began by observing that the 'neighbourhood' of each cell could be rotated around the centre in a limited way - by 90, 180 or 270 degrees. Pleasant symmetries were produced because no matter which of these 4 rotations (including the 'Identity', 0 degrees) is applied, the cell's state and the rules it followed would not change. In fact, researchers have made much of GL's symmetrical tendencies. It is worth mentioning that there was no 'toroidal' wrapping used during this research. The GL Universe is 'clipped' at the edge of the grid and any cell coming to life outside of the grid is discarded. The motivation for this was that a toroidal mapping reduces the effects of the radial symmetry - a toroid has toroidal symmetry instead.

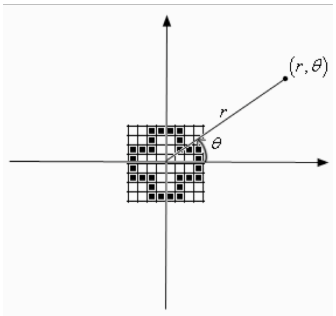


Figure 6: Polar diagram for the new radial mapping

4.2 Results

Thirteen initial seeds or states were allowed to grow for fifteen generations for all three mappings. Based on certain limiting assumptions about the application of Zipf's Law and Shannon's Information Entropy to music statistical tests were applied to measure entropy (a gauge of the amount of disorder). Figure 7 shows the results:

Mapping	Zipf R^2	Melodic Entropy	"Melodic" Zipf R^2
Radial	0.9	6.8	0.91
Cartesian	0.84	7.6	0.88
Binary	0.7	7.6	0.7
Random	0.55	9.7	0.54

Figure 7: *R2* Results of the various GL mappings for different initial seeds, 15 generations run.

The new radial mapping produced the highest 'melodic Zipf $R^{2'}$ and lowest 'melodic entropy' when compared against traditional Cartesian and binary mappings. Random 'music' gives the highest melodic entropy, which is compatible with it having the least note 'structure'. The lowest Zipf *R2* is given by the random music and the fact the radial mapping's Zipf *R2* score was the highest means it is a more musical mapping.

4.3 Discussion

'The evaluation supported the idea that the radial mapping provided some improvement.' [KM07] The point here is that by using a fresh paradigm for GL Kirke and Miranda produced a statistically significant advance in the field of mapping. What about cellular automata made with regular triangles or hexagons or with different rules? There may well be scope for further study with shapes that tessellate (due to the fact their interior angles are exact divisors of 360^{o}). Perhaps a CAAC system could be designed that changes the geometry or rules of the mapping in real-time? 3-D cellular automata are used in science to model brain tumour growth. Would it be possible to use three or even *n* dimensional shapes for CAAC generations?

5 Reflections

Research in CAAC systems is constantly evolving and scratching the surface of the available literature whilst studying this topic has been enthralling. McCormack's sophisticated work with multiple algorithm types characterizes the rigorous approach taken by many. Kirke and Miranda highlight the importance of the mapping element of a CAAC system and the author found their results intriguing.

Is it desirable to have a system that can make music without human intervention? Or is the best solution for man and machine to work as synergetic partners? Surely composers should still have to possess musical talent? A CAAC can be just that – a Computer *Aided* Algorithmic Composition system. Ready to help out and spark artistic inspiration.

The human-machine feedback loop is known as *computational synergetics* and it is an area of great interest to the author as it is here where many CAAC systems fail to capitalise on their potential. In the author's opinion this is largely due to a poor GUI coupled with lack of thought on *how* every useful idea can be captured. Perhaps we focus too much on asking *if* CAAC systems can compose and not enough time considering *how* we can best use them as an inspirational source. At the end of the day our common goal should be to make good music, intuitively.

The penultimate section that follows outlines a GUI for a CAAC system and attempts to describe how the interface could operate.

5.1 An Original Interface

This CAAC system should be as cross-platform as possible. OS X, Linux and Windows are fundamental and deployment on devices like the iPhone or iPod Touch would be desirable. There are so many mathematical possibilities for the algorithm and mapping that they cannot all be discussed here and would be best verified by experiment – both that of the CAAC system designer and the end user. One possibility for the internal algorithm would be a rule-based system, similar to that of William Schottstaedt where

around seventy-five rules derived from Fux's Gradus ad Parnassum were implemented. The author has begun work on the device using C and some machine code though Python may be a suitable language as it is renowned for being easy to learn, compact and intuitive.

The more readily ideas can be realised the greater the commercial benefits for the system vendor and the creative gains for the artist. Burraston [Bu05] describes a system where many different applications are needed before a musical idea can even be heard, saving and exporting files between software is required. This is symptomatic of many CAAC systems and risks losing inspiration. The following design is theoretical only but is intended to operate in real-time and crucially, in one environment. The goal here is to create an intuitive front end, capable of manipulating internal representations beneath which the core of the CAAC system operates.

One problem with existing CAAC systems is their level of user friendliness. It is beyond the scope of this report to discuss GUI's of CAAC systems with which the author is familiar but it can be said that many require knowledge of music theory above and beyond the average computer user. Terminology and symbols can seem intimidating and get in the way of composition. This system is designed with the modern Digital Audio Workstations (DAWs) in mind and would probably operate as a Virtual Studio Technology (VST) 'plugin'. In this way it would have access to sound sources within the DAWs environment (via routing) and so no sound generative device has been conceived, nor is it needed (although it could come later). The heart of the device is the 'piano roll', ubiquitous in modern music software applications.

5.2 An Overview

Although a mode for indirect representations such as a list of phone numbers could be incorporated, the author is mainly interested in the CAAC system manipulating direct representations in the form of MIDI or audio. All the information will be extracted from files imported (via a browser or drag and drop) into the main window. Behind the scenes one or more algorithms will use mathematical ideas and the music theory of intervals, chords, inversions, part harmonisation, concords, discords, chord progressions, cadences, counterpoint and so forth to generate data.

Whenever the device is operated, a recording will be made to the computer's hard drive. From this recording the composer would be able to select the usable forms, erasing the unsuitable ones. However, *much more usefully*, these usable forms can be dragged by mouse selection and dropped from the main window into the surrounding audio bins for later use. Although many different levels of hierarchy present in a composition could be manipulated, the illustration (figure 8) features a four beat bar split into parts. Notes occurring in the first quarter might need to be restricted to a certain chord, the second quarter may demand another chord and so generated patterns would conform to these requirements, throughout the chord progression.

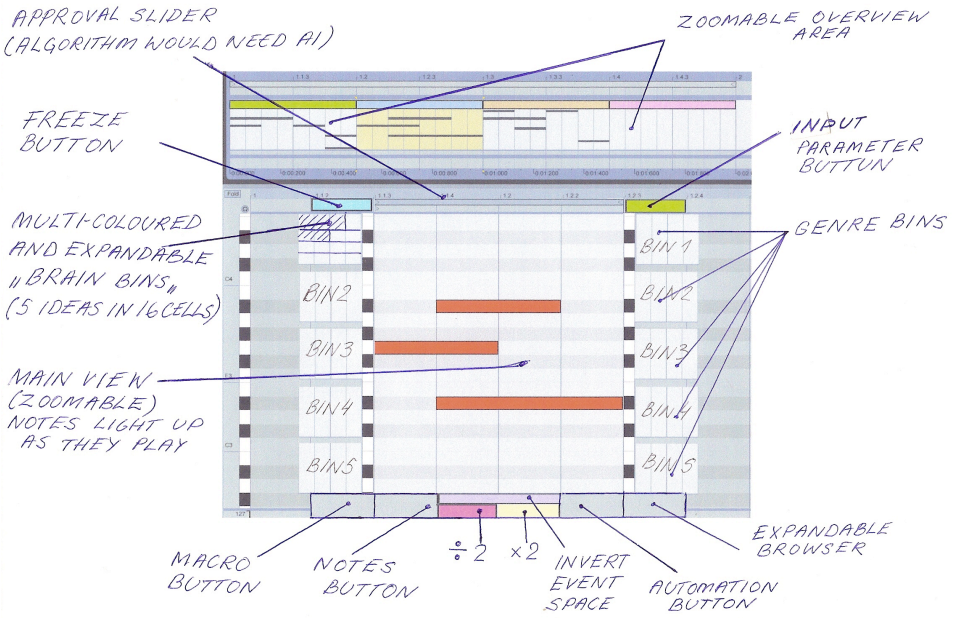


Figure 8: The author's GUI for a CAAC

The CAAC will ideally be able to import both mono and polyphonic audio files, detect their transients and notes to enable conversion to a MIDI file – when the technology exists to do so. Whilst MIDI to audio is trivial and omnipresent in DAWs, audio to MIDI is very much a work in progress. Some companies claim to have products that will transcribe a polyphonic source. The author is dubious. However, it may be possible to get the *key* of the audio and the author has had great success with a commercial key-detecting program called *Mixed in Key*. Knowing the key of an audio file would help and at least give the range of notes that could be used. A full MIDI representation would need to be used for harmonisation, chord progressions, counterpoint, cadences and so on. Once imported, the original MIDI (or audio converted to MIDI) would be displayed in the overview at the top. Generated data compatible with the section of the original file selected would flow through the main view window according to the parameters setup in the preferences.

5.3 The Controls

Some components of the GUI have already been mentioned and others should be self-explanatory. Features worthy of special mention are outlined in the following paragraphs. The Parameters button will provide access to the preferences allowing for example selection of non-Western scales or deciding which combinations of notes to avoid or favour. The Approval Slider is a negative to positive left to right 'never ending' scale that would demand Artificial Intelligence to utilise fully. The basic idea being that as the material flows through the main window, the user can show whether they approve of the music. The CAAC system could then act accordingly.

The Freeze button would freeze the flow and allow the user to manipulate the data in the main window, and the data before and after it, if they hear a particularly catchy motif (this is possible as the rendering of data is done instantaneously before being streamed through the main window.) The Expandable Browser would open up *both* compatible chord progressions, melodies, harmonisations and cadences etc and existing files on the computer that could be dropped into a bin or the overview window (and become the seed).

The idea of the bins is to store desirable material and whilst all data is recorded by the CAAC, the bins provide a way to capture and organise the music. Each bin is infinitely expandable as once four ideas are stored the cells become a 4 by 4 matrix giving space for 12 more ideas – all multicoloured to aid recall. The Macro button features customisable control over the inner workings of the algorithms and mappings – for example Markov probabilities. This will also allow an external MIDI device to be assigned to those parameters. The buttons for Notes and Automation (for example stereo panning) depict which form of MIDI data is visible in the Main View window. The divide and multiply by two buttons shorten or lengthen the bulk of the notes flowing through the Main View window. It is possible to hear any combination of Overview window (seed) and Main View window (generated data) simultaneously. Finally the Invert Event Space button is intended for freeze mode but requires a little explanation. 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*. The 'Invert Event Space' button in this design would actually remove the existing notes and fill the previously empty temporal space with notes.

5.4 The Future

In the above section AI was mentioned in the context of the Approval Slider. If and when computers possess the necessary 'Intelligence' to realise such notions is a matter of debate. Can a computer learn each individual user's unique personality, procedures, habits, and preferences, and to use this knowledge to take initiative and make musical suggestions?

In the more foreseeable future current technologies such as touch screen may evolve and become commonplace, defining how Multimedia Systems are designed. Already Apple's iPhone/ iPod Touch has a multitude of music applications including at least one cellular automata implementation. The touch screen technology in Microsoft's Windows 7 will add many possibilities for CAAC systems and software in general.

What about an iPhone/ iPod Touch tool that turns pictures into music? The iPod's accelerometer is also intriguing for music technology designers.

The recognition of hand gestures (gestures have both continuous and discrete information) may play a big role in CAAC systems of the future. Researchers at Boston's Northeastern University have designed an interface that can track the movement of the users hand in three dimensions.

In 2008 scientists reported on 'geometrical music theory' that translates the language of musical theory into that of contemporary geometry. They take sequences of notes, like chords, rhythms and scales, and categorize them so they can be grouped into 'families'. They have found a way to assign mathematical structure to these families, so that they can be represented by points in complex geometrical spaces. Perhaps this has implications for CAAC mappings? The future sounds bright.

Bibliography

[Bu05] Burraston, D.: Composition at the Edge of Chaos. In Proceedings of the 2005 Australasian Computer Music Conference, 2005

[Da57] Darreg, I.: The Electronic Computer Takes up Music Lessons. 1957.

[FS01] Farbood, M.; Schoner, B.: Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains. In Proceedings of the International Computer Music Conference. San Francisco: International Computer Music Association, 2001

[KM07] Kirke, A.; Miranda, E. R.: Evaluating Mappings for Cellular Automata Music In MusicAL: Workshop on Music and Artificial Life, 2007

[Ma00] Marsden, A.: Representing Musical Time: A Temporal-Logic Approach, Swets & Zeitlinger, Lisse, 2000.

[Mc96] McCormack, J.: Grammar Based Music Composition In Complex Systems 96: From Local Interactions to Global Phenomena. R. Stocker, ed. Amsterdam: ISO Press, 1996