Thoughts on Interactive Generative Music Composition

Samuel J. Hunt, Tom Mitchell and Chris Nash

Department of Computer Science and Creative Technologies
The University of the West of England.
samuel.hunt@uwe.ac.uk

Abstract. The ability for machines to compose novel, interesting and human-esque music is an important and recurring theme in the history of computer music. Modern staff notation remains the most common tool for music composition, and despite the emergence and abundance of highly flexible computer-based score editors, efforts to incorporate generative music techniques into such systems and subsequent composer workflow are seldom explored. This paper explores and develops both theoretical and practical models that address these concerns, and present these in the context of a software prototype. The paper is constructed as a work in progress, detailing the work so far and how future research will be conducted.

Keywords: Interactive Music Systems, Generative Music, Score Editors.

1 Introduction

Generative music, automated composition and algorithmic music are interchangeable terms referring to a formal process in which music is made with minimal human intervention [1]. Wooller et al [2] define generative music as a process whereby the output of some operation has more general musical predisposition than the input data (e.g. parameters) and the size of such data (e.g. notes) has increased. Generative music is used primarily in one of two contexts:

- For live music performance, where each recital is different to the next, for example, Terry Riley's in C [3].
- For a fixed composition, where generative processes are used to create a static composition, performed by musicians that is the same for each recital, for example Melomics [4].

The system discussed here is the latter type.

Contemporary generative music systems and languages often require the expression of generative algorithms using programming language syntax, a skill that must be developed by composers in addition to western score notation. For example, Sonic Pi [5], Impromptu [6] and Supercollider [7] to name a few,

provide tools supporting generative music but require expert knowledge of the system and require a transition into a different style of composition workflow. Many other systems such as Neural Networks and Evolutionary techniques often require the user to be competent with the underlying mathematical models, before being able to utilise them as tools for music composition, these systems are designed primarily for computer scientists, not musicians.

Open Music [8] is an environment for music composition that uses visual programming languages for the generation or manipulation of musical material. This provides a lower entry threshold than more traditional programming languages, and also allows familiar items such as scored elements to be retained. Nash's [9] Manhattan programming language extends a music tracker environment with spreadsheet based programming, therefore allowing the user to experiment with rule based and generative music inside an existing workflow. Following on with similar ideas, the aim of the research here is to extend score editor workflows with generative music tools.

Before designing a system that integrates interactive generative music with score notation, three overarching problems must be discussed, and these are:

- How can the various automated techniques for generating music be categorised?
- How can the workflow of using interactive generative music be modelled?
- How can the overall interaction and choices made by the user be measured and compared?

The central aim of the paper is to propose a series of ideas that can integrate generative music techniques inside score editor workflows, through proposed software prototypes.

In the sections that follow, a brief summary of existing generative techniques and systems is followed by a discussion on how such techniques may be categorised. Section 3 describes how identified problems with existing systems can be remedied through both closer integration with the composer's workflow, and by improving the interactivity for each generative technique. Section 4 discusses a model designed to capture the process of generative music, exploring briefly how this aligns with existing research into algorithmic composition systems. A prototype software implementation is discussed in section 5, and section 6 explores how such systems can be evaluated.

2 Generative Music Systems

A number of techniques and systems exist for generative music, however only a subset is relevant to this research. For example systems that incorporate supervised learning methods (for example neural networks [10, 11]) require extensive training before use. Systems such as rule based and stochastic models [12] are easy to comprehend and implement, as well as being fast to run, but can be criticised for producing simple music. Evolutionary techniques for producing music [13] on the other hand can produce sophisticated work, but require greater

knowledge and take time to compute, requiring the user to guide the generative process [14]. For fuller treatment on the vast use and history of generative music systems see [15, 16] and [17].

A prominent interactive generative music system that works in unison with a composer is the Continuator system developed by Pachet [18]. The Continuator uses Markov models to learn musical structure through either existing material or in real time from a performer, and can then reproduce music in either a standalone mode or in collaboration with the user. The system is intended for performance rather than traditional composition, but it does however conclude that the underlying complexity of using Markov models for composition can be made accessible for musicians.

3 Categorising Generative Music Techniques

Although many techniques exist for generative composition, it is difficult to appropriately group and categorise them. Wooller et al [2] define a framework for comparing algorithmic music processes that allows different systems to be related and contrasted. They group the techniques into three categories:

- Analytic: Reduce the size of input (or training) data to extract specific features, for example obtaining an appropriate set of notes from a database of sequences.
- Transformational: Transform a certain structural element of the music, for example modifying pitches without altering rhythm.
- Generative: Produces music from input data, or rules, resulting in an increase in data size, for example a chaos music algorithm utilising a single numbered seed to produce a sequence of randomised notes.

These elements are also considered in terms of their *contextual breadth*. Wooller et al [2] define this as the size of the surrounding data that has influence over the computation of the algorithm, this can be directly related to the size of the training set in a machine learning application for example. Applying retrograde, a type of transformational algorithm, takes an input sequence and reverses the play order. This type of process has no contextual breadth, whereas a pitch quantization function requires knowledge of the tonal context (key and scale).

This research looks to extend Wooller's [2] framework by adding an interactivity dimension, something overlooked in the original. The first type of interaction introduced for this research is One Shot Interaction (OSI), whereby the output is the result of pushing-a-button, for example using musical dice games [19] or simple musical transformations (Inversion, Retrograde, Transposition). These types of process create all the musical material in one step, without further user intervention. The second type is defined as Continuous Feedback Interaction (CFI), whereby the generative applications parameters are continuously updated either by the user or feedback from earlier processes. Examples of this include arpeggiators, the Continuator [18] and certain types of supervised learning systems [20].

CFI techniques can be seen as more interactive than OSI techniques, however CFI techniques are more challenging to integrate due to their inherit additional interaction complexity.

4 Problems with Generative Music

Summarizing findings in existing generative music research revealed several issues with the musical quality. Todd [21] notes that generative programs can easily get stuck in cyclical loops. Existing theories as observed by Pearce [22] focus on certain musical elements in isolation, whereas music in reality is a complex combination of these elements. Pearce and Wiggins [23] suggest that some computational models often fail to meet the intrinsic stylistic constraints of the genre. The computer can easily make music that hits the extremes of ranges, or groups notes all in the same register [24], therefore becoming inappropriate for the instrument, or performer. Finally, many of the systems reviewed are isolated away into scientific research projects [4] without conforming to existing compositional practice, requiring domain specific knowledge [25] and competency with the notation [6].

The most prominent issue within current systems is the lack of high-level structure [11, 14, 24]. A solution to this and other issues previously discussed is to introduce the human as an active agent within the system [6]. Creating a system that acknowledges the strengths of both the human composer and the generative process, enables an effective collaboration to take place. The aim is not to replace the human composer, but rather to aid them. Furthermore Myhill [26] argues that algorithmic music should have both deterministic and stochastic elements without the loss of stylistic coherence, therefore it is suggested in this project that the human composer can control the style and deterministic elements with the computer providing the stochastic elements.

5 Modelling Generative Music

An important consideration for exploring generative music, is how the interactive process and workflow is modelled. This research utilizes the ideas presented by Logic Pro X's [27] smart drummer, in that a piece of drum music is divided into generative sections. Each of these sections has a set of parameters that are evaluated to create music content that remains the same until a parameter is changed. The core idea being that the notes themselves are not specified but rather the parameters and techniques used to create them (Figure 1). For example in Figure 1 the amount and complexity of the hi-hat pattern used in the section is controlled by a single slider. The overall pattern is controlled by a 2D grid through specifying the amount of Complexity vs Simplicity and Loud vs Soft (left in Figure 1). Incorporating these ideas here requires the addition of an option that will lock or store the previous iteration so no further iterations are evaluated.

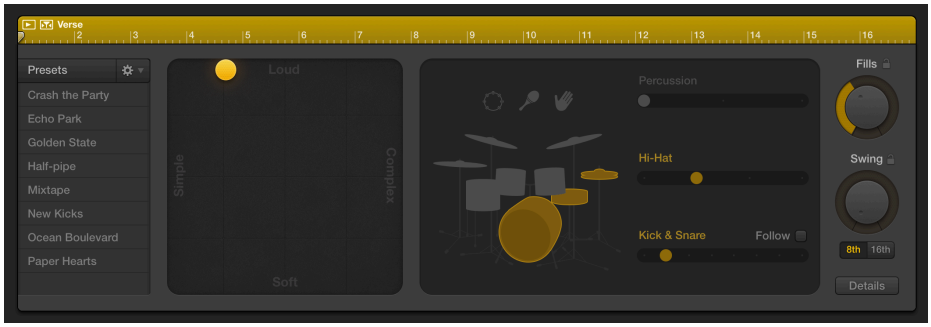

Fig. 1. Logic Pro X's [27] Smart Drummer plugin.

5.1 Iteration Management

A further consideration for this work is how each iteration is stored. A user may create many iterations of music formed through the model, only to find the most suitable has been lost. To address this, the system must make use of source/version control techniques. Providing both a method for capturing un-reproducible music parts, and the ability to recall generative interactions in a list, facilitating and encouraging musical experimentation. Finally Duignan, [28] notes that version control and snapshot tools are practically non-existent in existing music software, noting that users have to work around such issues, this therefore suggests music software systems could benefit from such tools.

5.2 Creativity

Formal and generalised models for capturing a generative composition workflow have little exploration in current literature. There is however some more notable work, modelling generalised music composition. For example, Alty [29] presents work in which the composition process is modelled as a journey through a series of state spaces, with each state representing a possible option in which the musical direction can take. Constraints are often applied to the states, limiting the choice of direction and morphing the task into a problem solving exercise. Such exercises are prime candidates for solving with a variety of different computer models and algorithms.

A more general issue around creativity is that there is no one solution for supporting creative tasks. Creativity by its very definition results in a wide variety of different workflows and approaches. The models discussed in this paper are only an example way of working, acknowledging that other users may prefer a yet unknown model or solution, more suited to their own workflow.

Finally, Jacob [30] draws concern over the authorship of music composed by algorithmic methods, attributing the music to the designer of the algorithm not the user of it. His solution to avoid this criticism is to implement one's own algorithm. However, the focus in this project is on the arrangement, interaction, and combination of different techniques, not the design of the technique itself.

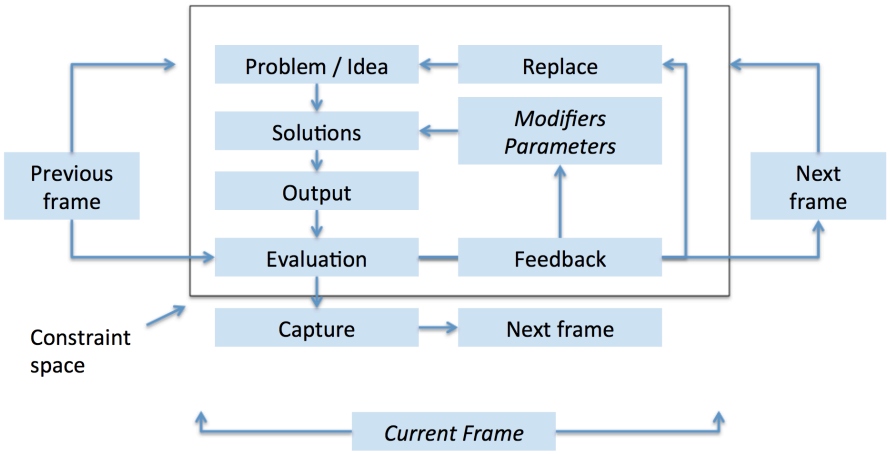

Fig. 2. Prototype model of generative music interaction.

5.3 The Model

Figure 2 shows the prototype model used in this research for defining generative music as a composition process. The *problem/idea* is defined as the input material, this can be a collection of pitches, or a sequence (time series) of notes. The process takes the Input and modifies it based on a series of rules or transformations to create solutions, with certain controlling parameters exposed for live interaction. Following on from this is the *output*, at which point the music becomes audible. The *evaluation* stage creates a branch, where the user is either satisfied with the output, or provides feedback to the initial seed and parameters in the process stage. This loop continues until the *evaluation* stage is complete, at which point a capture is created, and the next frame begins. The constraint space [29] governs restrictions placed upon the composition, for example using the key C-major makes certain choices of pitch more likely [31].

The previous frame has influence on the current frame in that it can impose restrictions over the constraint space. The evaluation stage is considered in reference to the previous frame, as music is auditioned in respect to a timeline. Likewise, changes made from the feedback stage in the current frame have a knock-on effect for the next and subsequent frames. A frame has no defined length, but in general should be considered something that is more than a single note, but less than an entire section. A key consideration that differentiates this model from other generative music systems, is that each frame of music is carefully controlled and unlikely to spiral into chaos, but retains inter-frame influence (higher level musical structure) through attentive links between frames.

An arpeggiator is a simple transformation based music generation technique that can be easily mapped to the above model. The input *(problem/idea)* is the

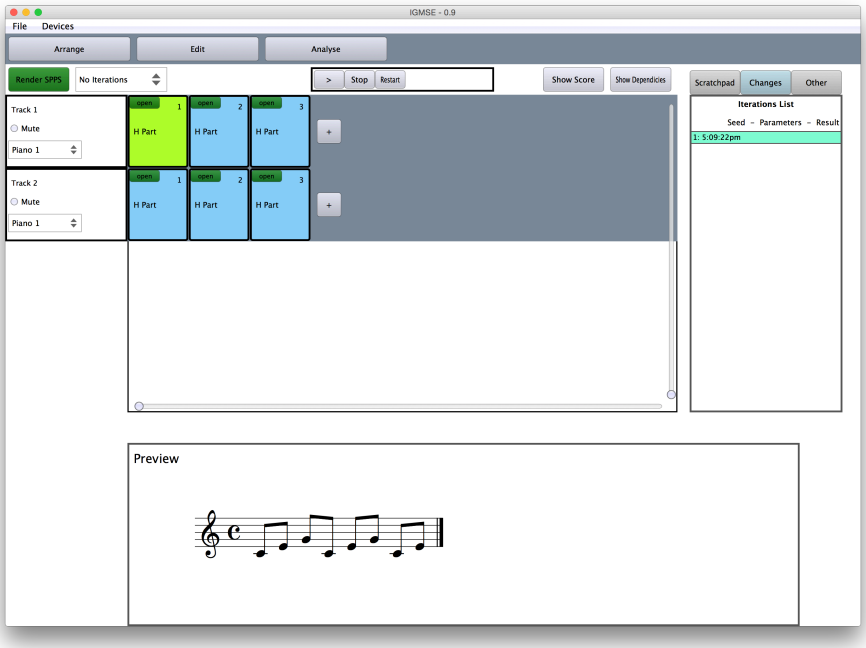

Fig. 3. Prototype Software showing how the individual blocks are arranged (arrange view).

notes of the arpeggiator, the *parameters* represent the speed, octave and rhythmic pattern controls. The net result of the input notes and specified parameters produces an output. Upon hearing the output, the user may either update the input notes, or arpeggiator parameters which in turn creates different output material. Once the user is happy then a *capture* is created for a given number of bars. The next frame after this is influenced by the previous frame in that certain parameters are likely to be shared between the two. The constraint space in this example is governed by the limitations of the arpeggiator effect, for example the arpeggiator may only work with a total of 6 input notes at a time.

5.4 Existing models

The model proposed here is similar to existing research in modelling different music composition processes. For example Papadopoulos et al [32] describe algorithmic composition as a set of rules for solving a problem of combing musical parts into a whole composition. Relating the proposed model to this statement, the *frames* are each of the musical *parts* that form the composition. The set of *rules* in their work relate to this model's *constraint space*.

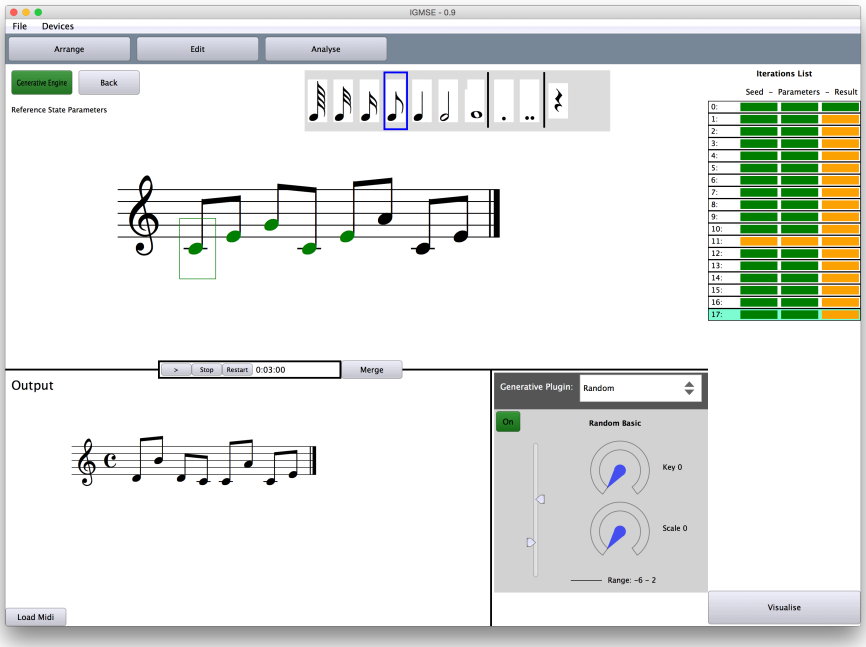

Fig. 4. Prototype Software showing how the input material, output material and generative parameters are interacted with.

6 Software Implementation

The prototype software displayed in Figures 3 and 4 show the arrange view and edit view respectively. The first is responsible for arranging the individual parts (blocks) of music, the second is for editing the individual blocks. Both of these views relate to existing computer music workflows (e.g. Digital Audio Workstations).

Figure 4 shows the edit view, the entry editor (top) shows the standard western score the user interacts with for composing an initial piece of music. In this example, the generative technique used is a simple random note generator, whereby green notes illustrate notes that will randomly update on each iteration, by an amount specified by the parameters in the bottom right. Each iteration is listed in the table on the right hand side, storing the seed, parameters and result (green highlights no change, orange highlights a change). The output from the current (or selected) iteration is shown on the bottom.

Each iteration of the algorithm can be computed as a recursive or non-recursive system. In a non-recursive state, the initial seed is simply re-evaluated each time, producing a finite amount of unique outputs. A recursive setup means that the output becomes the seed, and subsequent iterations derive sequentially

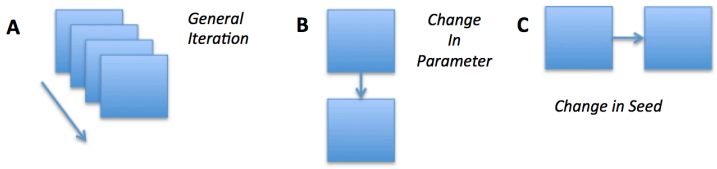

Fig. 5. A: Iterating through the generative model, B: Changing generative model parameters, C: Changing the initial input material (seed).

from each other. A negative effect of this means that notes can slowly progress towards the extremes of each register, meaning that wide intervals between notes become more likely. This is in general not in-keeping with analysis conducted of existing music [31]. It is however important to keep these options open to users, allowing them to decide whether or not they are suitable.

7 Evaluating Generative Music Systems

Although there are many specific goals for music composition, the primary consideration for evaluating generative music in this situation is measuring whether the output is suitable for the composition task, as defined by the composer. During music composition, the composer will likely try out many different musical ideas before ultimately committing to some final material. The generative music element is to help inspire ideas and solutions to these musical problems. Evaluating the musical quality of work produced by this system is not currently a focus of this research.

The current goal of the system is to measure how users interact with the software to find suitable music solutions. For example a list of all changes and iterations (version control) are stored (as shown in Figure 6), therefore the resultant musical journey can be analysed to find patterns. For example composer A using technique 1 used 50 iterations before finding suitable material, whereas the same composer using technique 2 needed only 5 iterations. This will be contrasted with other methodologies such as using questionnaires with Likert scales to subjectively evaluate qualitative opinions, for example *"I found this technique to be enjoyable/frustrating"*. At this stage it is unclear whether lots of iterations within a generative music model is correlated with enjoyment (enjoying listing to the different iterations) or frustration (taking to long to find a suitable solution).

In general the three observable variables in this model are changes in seed (initial musical material), changes in model parameters and the number of iterations. This can be visualised (Figure 5) using a three-dimensional grid, where changes in X represent changes in seed (Figure 5-C), changes in Y (5-B) represent changes in parameters, and changes in Z or layers (5-A) representing iterations (changes in each of the dimensions are shown in Figure 5). The diagram in Figure 6 shows a specific configuration, but in reality, changes in X, Y and Z can

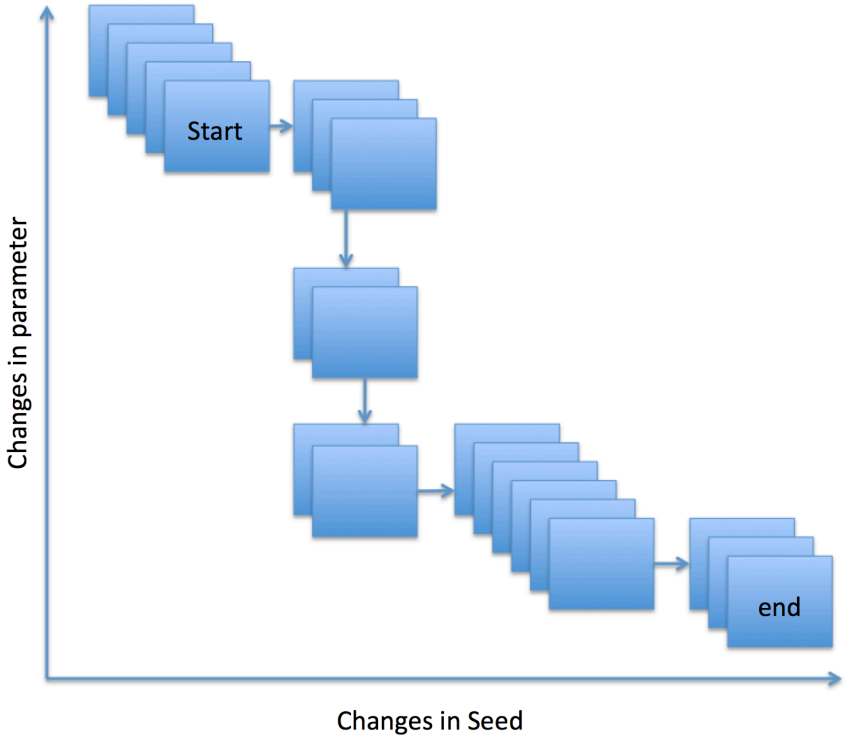

Fig. 6. A visulisation example of the changes made by a user.

happen in any order. A useful side effect of this configuration is the choices made can be visualised to produce a quick overview.

Gathering the choices made by users when interacting with the proposed software is easy accomplished by interaction logging. Brown et al [33] summarise the evaluation techniques used within the NIME and ICMC conference proceedings, and note that interaction logs are often used to measure usability. Nash [34] also notes that as interaction logging is non-invasive it enables the study of real-world creativity, without interfering with the individuals creative process or intruding in their environment.

In summary the technique discussed in relation to Figure 5, interaction logging techniques, and qualitative user surveys will be used in conjunction with one another to evaluate and draw conclusions around this proposed system. A series of pilot of studies are also planned, to showcase the software and its workflow to professional composers.

8 Conclusions

This paper sets out initial thoughts and arguments for integrating generative music techniques inside a score editor workflow. The next step is to extended the software prototype shown in Figures 3 and 4 so that it can be given to a series of pilot users. Initially only using a small number of simple generative techniques (for example random, rule based and Markov models) to observe primary findings in usage patterns. The integration of more complex techniques (neural networks and evolutionary techniques) are reliant on positive results in using these more simple techniques. Overall this research aims to make generative music techniques more accessible to composers whose primary software is rooted in score editor paradigms.

References

1. A. Alpern, "Techniques for algorithmic composition of music," *http://hamp.hampshire.edu/adaF92/algocomp/algocomp*, vol. 95, p. 120, 1995.
2. R. Wooller, A. R. Brown, E. Miranda, J. Diederich, and R. Berry, "A framework for comparison of process in algorithmic music systems," in *Generative Arts Practice 2005 - A Creativity and Cognition Symposium*, (Sydney, Australia), University of Technology, Sydney, Creativity and Cognition Studios, 5-7 December 2005.
3. K. Potter, *Four Musical Minimalists: La Monte Young, Terry Riley, Steve Reich, Philip Glass*, vol. 11. Cambridge University Press, 2002.
4. C. S. Quintana, F. M. Arcas, D. A. Molina, J. D. F. Rodríguez, and F. J. Vico, "Melomics: A case-study of ai in spain," *AI Magazine*, vol. 34, no. 3, pp. 99–103, 2013.
5. S. Aaron and A. F. Blackwell, "From sonic pi to overtone: Creative musical experiences with domain-specific and functional languages," in *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling and; design*, pp. 35–46, ACM, 2013.
6. A. Sorensen and H. Gardner, "Programming with time: Cyber-physical programming with impromptu," *ACM Sigplan Notices*, vol. 45, no. 10, pp. 822–834, 2010.
7. S. Wilson, D. Cottle, and N. Collins, *The SuperCollider Book*. The MIT Press, 2011.
8. J. Bresson, C. Agon, and G. Assayag, "Openmusic: Visual programming environment for music composition, analysis and research," in *Proceedings of the 19th ACM International Conference on Multimedia*, pp. 743–746, ACM, 2011.
9. C. Nash, "Manhattan: End-user programming for music," in *14th International Conference on New Interfaces for Musical Expression*, (Goldsmiths, University of London), NIME, 30 June 2014.
10. K. Choi, G. Fazekas, and M. Sandler, "Text-based lstm networks for automatic music composition," in *Conference on Computer Simulation of Musical Creativity*, (London, England), Queen Mary, arXiv, April 2016.
11. M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.
12. F. Brooks Jr, A. Hopkins Jr, P. G. Neumann, and W. Wright, "An experiment in musical composition," *Electronic Computers, IRE Transactions on*, vol. 6, no. 3, pp. 175–182, 1957.

13. J. Biles, "Genjam: A genetic algorithm for generating jazz solos," in *Proceedings of the International Computer Music Conference*, (Denmark), pp. 131–131, International Computer Music Accociation, Michigan Publishing, September 1994.
14. G. Wiggins, G. Papadopoulos, S. Phon-Amnuaisuk, and A. Tuson, "Evolutionary methods for musical composition," in *AISB Symposium on Musical Creativity*, (Jyväskylä, Findland), University of Edinburgh, 1998.
15. D. Cope, *Computers and Musical Style*. Oxford: Oxford University Press, 1991.
16. C. Ariza, *An Open Design for Computer-Aided Algorithmic Music Composition*. Boca Raton, Florida: Universal-Publishers, 2005.
17. J. D. Fernández and F. Vico, "Ai methods in algorithmic composition: A comprehensive survey," *Journal of Artificial Intelligence Research*, vol. 48, pp. 513–582, 2013.
18. F. Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.
19. L. Harkleroad, *The Math Behind the Music*, vol. 1. Cambridge University Press Cambridge, 2006.
20. A. E. Coca, D. C. Corrêa, and L. Zhao, "Computer-aided music composition with lstm neural network and chaotic inspiration," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, (Dallas, Texas), pp. 1–7, IEEE, IEEE, August 4-9 2013.
21. P. M. Todd, "A connectionist approach to algorithmic composition," *Computer Music Journal*, vol. 13, no. 4, pp. 27–43, 1989.
22. M. T. Pearce, "Early applications of information theory to music," 2007.
23. M. T. Pearce and G. A. Wiggins, "Evaluating cognitive models of musical composition," in *Proceedings of the 4th international joint workshop on computational creativity*, pp. 73–80, Goldsmiths, University of London, 17-19 June 2007.
24. A. C. Sorensen and A. R. Brown, "A computational model for the generation of orchestral music in the germanic symphonic tradition: A progress report," in *Proceedings of Sound : Space - The Australasian Computer Music Conference*, (Sydney.), ACMA, ACMA, 10-12 July 2008.
25. C. Nash, "The cognitive dimensions of music notations," in *Proceedings of the International Conference on Technologies for Music Notation and Representation - TENOR2015*, Institut de Recherche en Musicologie, 2015.
26. J. Myhill, "Controlled indeterminacy a first step towards a semi-stochastic music language," *Computer Music Journal*, vol. 3, no. 3, pp. 12–14, 1979.
27. K. Anker and O. Merton, *Logic Pro X Power!: The Comprehensive Guide*. Cengage Learning PTR, 2014.
28. M. Duignan, "Computer mediated music production: A study of abstraction and activity," *Computer Music Journal*, vol. 34, no. 4, pp. 22–33, 2010.
29. J. Alty, "Navigating through compositional space: The creativity corridor," *Leonardo*, vol. 28, no. 3, pp. 215–219, 1995.
30. B. L. Jacob, "Algorithmic composition as a model of creativity," *Organised Sound*, vol. 1, no. 03, pp. 157–165, 1996.
31. D. Temperley, *Music and probability*. The MIT Press, 2007.
32. G. Papadopoulos and G. Wiggins, "Ai methods for algorithmic composition: A survey, a critical view and future prospects," in *AISB Symposium on Musical Creativity*, (Edinburgh, UK), pp. 110–117, 1999.
33. D. Brown, C. Nash, and T. Mitchell, "A user experience review of music interaction evaluations," in *17th International Conference on New Interfaces for Musical Expression*, (Aalborg University, Copenhagen.), NIME, May 2017.

34. C. Nash, *Supporting Virtuosity and Flow in Computer Music*. PhD thesis, University Of Cambridge, 2011.