GeNotator: An Environment for Exploring the Application of Evolutionary Techniques in Computer Assisted Composition

Kurt Thywissen
Cacafoniq (www.cacafoniq.com)
kurtt@emu.com

Abstract

A computer assisted composition tool for investigating the application of evolutionary techniques in the composition of music is presented. The nature of such an application is examined in terms of defining the possible mechanisms that provide a means for automated creativity. These mechanisms take inspiration from processes found in Darwinian based evolution theory, genetic algorithm theory and similar aesthetically based uses of a genetic search heuristic in the visual arts. A formal model of "musical evolution" is proposed, with particular emphasis placed on the ways in which a genetic algorithm can be used to effectively manipulate a variety of compositional structures within a hierarchical and generative grammar- based model of musical composition. The result is a prototype Windows MIDI application called GeNotator that allows composers to experiment with a range of musical structures by interactively "evolving" them and do so through a familiar and comprehensive graphical user interface.

1 Introduction

Composing music is a creative process, and like any creative process, it can be described in terms of an aesthetic search through the space of possible structures that satisfy the requirements of that process: in our case, creating interesting music. Compositions tend to exhibit varying degrees of structure, where the composer labours by crafting and evolving initial ideas into satisfying end products. Richard Dawkins, in *The Blind Watchmaker* [Dawkins, 1986], describes lucidly how objects of apparent high structure (living organisms) can come about via evolutionary principles rooted in natural selection. He also extends this notion to a form of "Universal Darwinism" [Plotkin, 1995] whereby the creative generation of intellectual ideas themselves derive from such a process of iterative refinement of competing ideas, or "memes" as he likes to call them.

The success of employing evolutionary techniques in artistic endeavours is best exemplified by the work of William Latham and his *Mutator* and *Form Grow* systems for evolving 3D computer sculptures [Latham *et al.*, 1992]. His results can only be described as extraordinary, surreal and distinctly organic. Latham observes, "The machine has given me freedom to explore and create complex ... forms

which previously had not been accessible to me, as they had been beyond my imagination."

Inspired by the work of Dawkins and Latham, *GeNotator* is an attempt at transferring these evolutionary concepts to the domain of music making. Its main "evolutionary" mechanism is the Genetic Algorithm (GA) [Goldberg, 1989], which has proven to be a very effective way of blindly seeking out acceptable candidate structures from large search spaces. In GeNotator, the GA is used to guide the composer through the search space of possible compositions.

GeNotator is currently a Windows9x/NT MIDI application written in C++ using the pubic domain Maximum MIDI toolkit. Additional components relating to the World Wide Web are implemented as ActiveX controls.

2 BACKGROUND

GeNotator is one of several previously reported systems that use so called "evolutionary" techniques in algorithmic composition [Biles, 1994]. Such techniques have also been used in sound synthesis applications such as parameter optimization for matching instrument designs [Horner, 1995]. Common to all of these systems is the Genetic Algorithm [Goldberg, 1989], the most widely used mechanism in evolutionary computation. For more on genetic algorithms, the reader should read Goldberg [Goldberg, 1989] as a good introductory text.

Briefly here, a GA is a probabilistic algorithm that maintains a population of individuals that encode parameters in the problem domain, typically for an optimization problem. The GA iteratively manipulates generations of such populations through the simple genetic manipulations *crossover* and *mutation*. By scoring chromosomes according to their performance, fitter individuals tend to eventually dominate a population as superior genetic content is allowed to evolve over time. In computer music the goal is to evolve aesthetically pleasing musical structures through interactive subjective fitness evaluation by users.

When considering a GA implementation the challenge is to find a representation scheme that maps a chromosome's makeup to musical features such that music can be "evolved". The objective fitness function is usually re-interpreted as *subjective*; the composer's taste and judgment instead dictate the relative success rate of competing musical structures represented by the chromosomes. As Biles [1994] observes, this is often the "bottleneck" in a GA-based system as GAs typically operate on relatively large populations of candidate chromosomes, each of which must be evaluated by the listener. One way to minimize this population explosion is to concentrate on defining higher level generative processes and/or structures that are in some way parametrically controlled; it is then these parameters which are evolved by the GA. These generative processes thus define boundaries to the composition form space, within which the GA can evolve interesting parameter combinations.

This is certainly the case with Latham's work [1992], where the user is typically given only 9 items to select between at each iteration of

evolution, yet each of those nine items already exhibit a high degree of pre-defined structure (as opposed to purely random starting conditions). The limitation with this approach is that the composer still needs a degree of analytical skill in deriving these generative processes/structures in the first place. However, once in place, anyone can theoretically act as "composer" and evolve music within the predetermined form space.

There is a second approach to tackling the population bottleneck, which is to somehow model salient features of the composer's taste itself, thus freeing him/her from having to laboriously judge a large population. One proposed method of doing this is to use a neural network to try to deduce a pattern of taste, trained by observing past user judgments [Thywissen, 1993, Biles, 1994]. Another approach is to assist chromosome fitness evaluation by allowing the user to define rules that, when satisfied, contribute to the chromosomes fitness score [Thywissen, 1993].

GeNototor is an example of the former approach in that it's higher level generative processes/structures are defined in terms of *generative music grammars*, not too dissimilar from linguistic grammars [Chomsky, 1957], a feature that differentiates this work from previously reported approaches. Much research into music-oriented grammars exists [Holtzman, 1981], and it is their formal descriptive, and generative power which is of particular interest here.

Complementing this generative music grammar infrastructure, GeNotator adds to the mix its own tailored version of the genetic algorithm.

3 OVERVIEW

GeNotator is an attempt to define a comprehensive framework for musical evolution. Figure 1 gives a high level view of its principal philosophy. In purely abstract terms we can picture a composition as an amorphous whole of inter-related parts where each plays a minor or major role in the composition's overall structure. For example, these parts may be a melody, a rhythmic structure, harmony, instrumentation or the form of a piece, to name just a few of the musical descriptors with which we are interested in manipulating. "Blind" composition involves applying the genetic algorithm to aspects within this amorphous whole such that permutations of the composition space can be evolved.

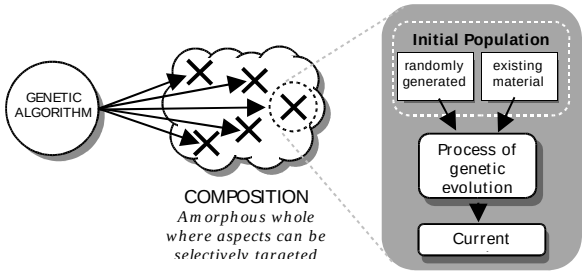


Figure 1 - Evolving different aspects of a composition

In terms of its place in the tradition of computer assisted composition, GeNotator is best described as a hybrid-algorithmic composition tool which, on the surface, resembles any mainstream MIDI-based compositional GUI environment providing tools such as piano-role and drum map editing, and an arrangement editor. However, to complement this is an extended framework, with GA at its heart, for evolving music.

Before going into detail about this framework, it is worth discussing briefly how GeNotator distinguishes between two different types of user/composer.

3.1 "Meta-composer" vs. "Gardener"

GeNotator operates at two distinct interactive levels, which are best classified in terms of user sophistication: the *meta-composer* vs. the *gardener* (to borrow terminology from Latham). Both, incidentally, could be the same person. The meta-composer tends to have a more analytical understanding about the form and structure of a composition and what he/she wants to achieve, and uses the GA to assist the creative process by generating and refining ideas.

Contrasted with this is the gardener who escapes from analytical thought entirely (is, in fact, not permitted to think in terms of structure definition), and instead only needs to know what he/she likes. This is borne from the observation that people tend to be far more sophisticated in listening than in creating music ("I don't know anything about music, but I know what I like").

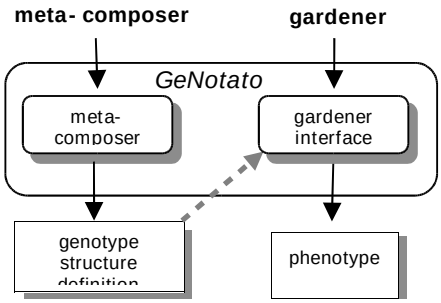
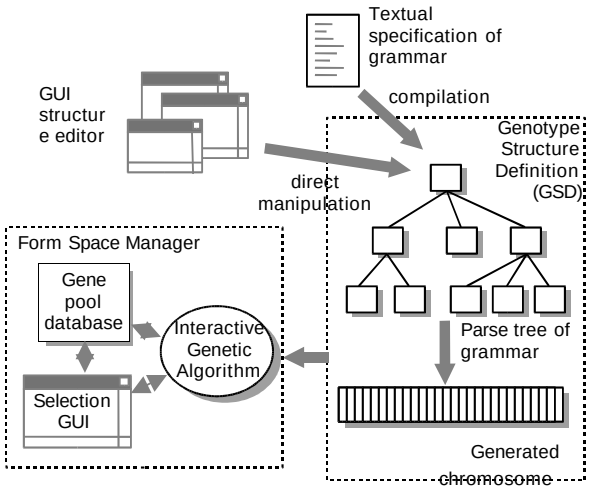


Figure 2 - Meta-composer vs. gardener

Both types of user have differing, and to some extent, complementary interfaces to a given composition. A typical scenario involves the meta-composer "publishing" a compositional structure genotype; the gardener is subsequently a consumer who interactively evolves a published meta-composition into any number of phenotypes – see figure 2.

3.2 GeNotator's Architecture

Figure 3 illustrates the main components in GeNotator. Central to this architecture is the *Genotype Structure Definition* (GSD).



The GSD is a data structure that packages a user defined music grammar together with an automatically generated genetic description that maps individual chromosome genes to choice routes in the grammar. The user specifies this grammar either textually or through a set of editing windows.

Once defined, the GSD serves as input to the *Form Space Manager*. This consists of an interactive genetic algorithm that takes the chromosome structure of the GSD and allows the user to breed and mutate phenotype instances of it. These are realized for playback and auditioning by compiling a MIDI stream using the grammar and the phenotype gene settings. The user is able to judge and score each on aesthetic merit and continue to do so iteratively in order to evolve favored instances.

3.3 GeNotator's Generative Music Grammar

A user will typically codify the structure of a composition via GeNotator's native generative music grammar scripting language. The syntax of this language resembles a familiar production/rewrite rule type notation. Aside from the basic constructs of iteration, concurrency and choice, the grammar has other more musically useful constructs allowing the composer to describe objects such as scales, keys, rhythms, phrases and larger compositional structures relating to a composition's "form".

Additionally, the grammar syntax is powerful enough to describe *transformational rules* as part of the grammar definition. This is an attempt to incorporate the linguistic notions of *deep structure* and *transformational rules* as described by Chomsky [Chomsky, 1956]. Simple examples of such transformational rules are: transposition, retrograde, inversion, key-change, canon and so on. These provide a very powerful generative way of creating additional variance in the

genotype description, and also allow the composer to describe music that has some sense of temporal evolution.

3.4 Specifying the Grammar through a Graphical User Interface

Up until recently, the grammar had to be specified almost exclusively in a text-based fashion - which is can prove prohibitive to those not versed in the fine art of programming. One of the main recent developments in GeNotator is the provision of an alternate *graphical* way of specifying all the available constructs in the grammar.

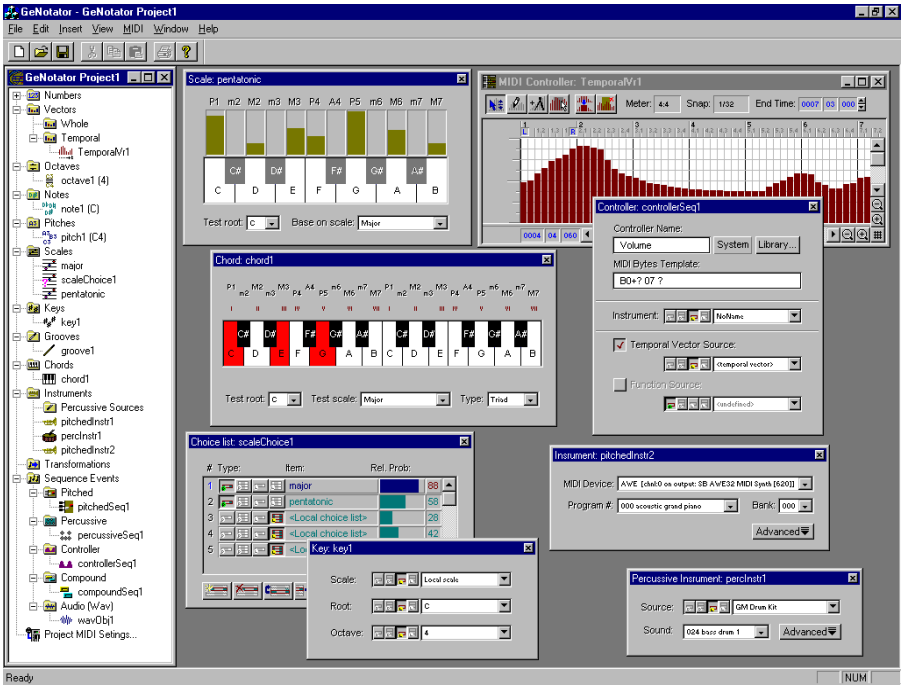


Figure 4 - Various GeNotator editing windows for constructing a grammar.

Figure 4 shows some of the editing windows that are now available for building scales, keys, chords and so on. Additionally, any given composition is organized as a *project* where each component in the project is typically a rewrite rule in the grammar. Since GeNotator has a set of familiar sequencer type editors (figure 5), and can import MIDI files, the result is an integrated environment which helps hide the fact that the composer is actually constructing a formal grammar of the composition behind the scenes.

Technically speaking, using the GUI for grammar specification does not result in the generation a textual description that is then compiled, but bypasses the compilation stage altogether. By interacting with the various editing windows the user is manipulating the internal parse tree representation of the grammar itself, an approach known as *structure editing*.

GeNotator actually permits the user to mix and match between a text-based grammar and the graphical approach within the same project. The GUI components can be seen as an alternative view of the grammar to that offered by the textual representation, and have the benefit of being easier to use for non-programmers - an important consideration if GeNotator is to be adopted by users not versed in programming (see figure 3).

3.5 From Grammar to Genotype

As mentioned earlier, the GSD consists of a grammar and a generated chromosome template. The grammar is stored internally as a parse tree together with an optional textual representation that was used to generate it. The complementary chromosome structure is automatically generated by a process of searching through the parse tree representation and introducing a new gene for each choice route expressed in the grammar. The allele cardinality of the new gene corresponds to the number of alternatives dictated by the choice route.

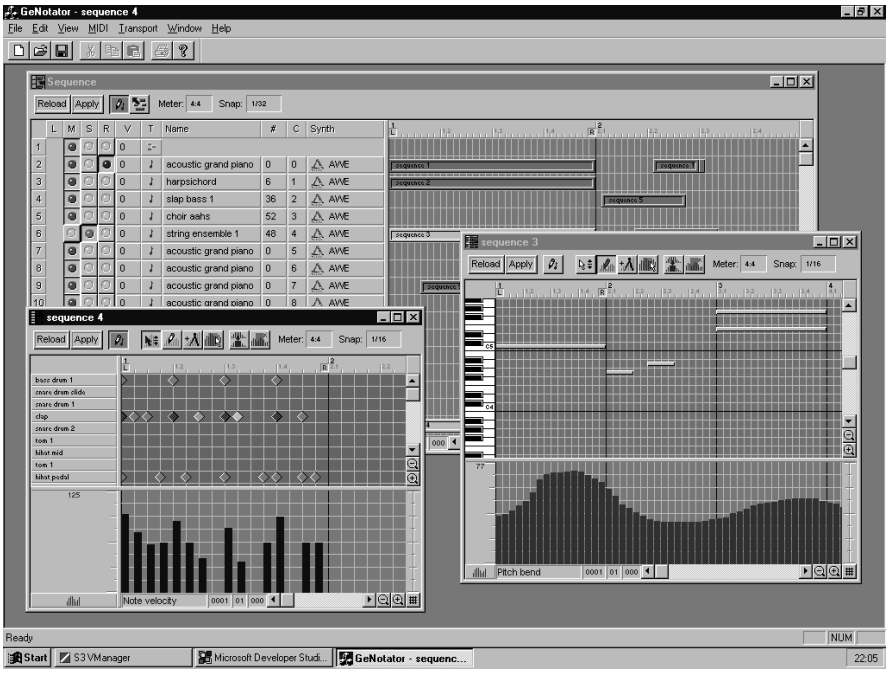


Figure 5 - Additional sequencer oriented editing features in GeNotator.

A simple example of this is shown in figure 6 below where one sequence rewrite rule is resolvable to three different alternate sequences, and the other one to two. In this instance two genes are added to the genotype with respective cardinalities three and two.

Since all the major constructs in GeNotator's grammar syntax support "choice", and thus variability, the meta-composer is free to define any range of variance desired within a composition genotype, from highly constrained to wildly unpredictable.

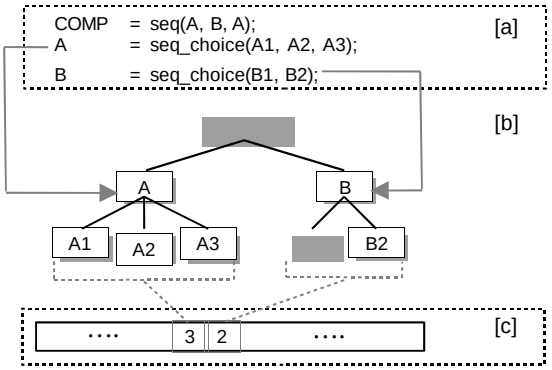


Figure 6 - A simple grammar [a] and compiled parse tree [b], with corresponding genes in chromosome structure

4 The Compositional Process

A typical scenario involves the composer specifying a meta-composition grammar textually or through the provided editors. A starting point may be an existing composition imported as a MIDI file, which the user can then split up into a grammatical structure. Or the composer may be attempting to formally describe a well-defined composition grammar for the purposes of proving the validity of particular ideas in music theory.

Depending on the amount of variance expressed in the grammar, the user can expect to be able to generate either a small or large variety of phenotypes from this grammar genotype, the search-space exploration of which is achieved through the FormSpace Explorer, which is the subject of the next section.

5 Evolutionary Mechanisms

GeNotator has its own "butchered" version of the genetic algorithm to implement the main evolutionary mechanism. Although considerably modified, this is in line with recent trends to move away from the prevailing classical binary model [Michalewicz, 1995].

As Michalewicz observes, "...to solve a nontrivial problem using an evolution program, we can either transform the problem into a form appropriate for the genetic algorithm, or we can transform the genetic algorithm to suit the problem."

5.1 A Tailored Interactive Genetic Algorithm

GeNotator is very much an example of this philosophy, and for this reason it is perhaps best to describe it as an *evolution program*. For example, although chromosomes are still of fixed length, each gene does not have to be binary, but may have its own cardinality independent of other genes. Additionally, it is possible to impose a probability distribution over the range of values individual genes may take during mutation to create bias towards a range of values.

Another major difference over classical approaches is the provision for an arbitrary number of parents beyond just two during genetic crossover.

The user interacts with this modified GA by controlling such factors as population size, mutation likelihood and designating which parents partake in breeding. By auditioning and scoring phenotypes, the user exerts influence on the direction the evolutionary process follows. Figure 7 opposite shows the user interface component that GeNotator provides for doing this, called the *FormSpace Explorer*.

5.2 Navigating Musical FormSpace

As figure 7 illustrates, the user navigates through the form space of a particular musical grammar by cultivating fit individuals through the iterative manipulation of individual populations. Individuals are presented as iconized faces, whose facial expressions indicate their relative worth. By double clicking on these faces, the user is able to listen and judge, at any time, the musical instantiation of the grammar permutation represented by that face. Right clicking on a face allows the user to score particular individuals, which in turn dictates the amount of influence these individuals will have on subsequent generations.

Controls are provided for setting the standard evolutionary parameters of population size, crossover and mutation probability. A third parameter, mutation sensitivity, is provided to control how drastic a gene may mutate over its cardinality range. Finally, a "scratch" area is provided within which the user can put aside favored individuals. Notice that the mating pool can contain more than two parents for any given generation allowing 3, 4 or *n* parents to generate the next population.

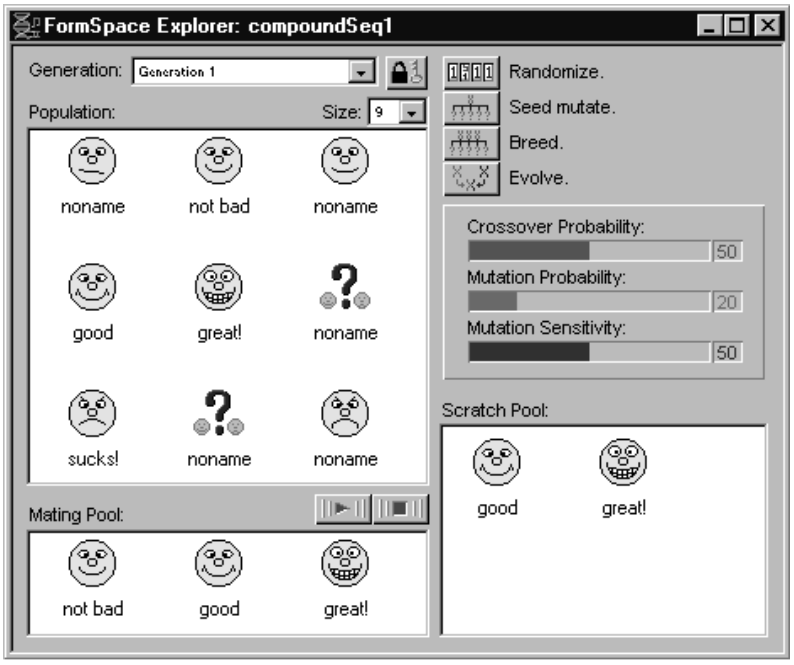


Figure 7 - Interactive navigation through "Musical FormSpace" with the FormSpace Explorer

5.3 The FormSpace Manager (FSM)

The FormSpace Manager (FSM) is where GeNotator's tailored GA resides along with other house-keeping facilities such as a gene-pool database used to maintain and retrieve generations of genetic content. Additionally, the FSM offers a hierarchic menu interface that permits users to progressively unlock deeper structural content in a given GSD.

In practice, this means facilitating a way for the user to target particular aspects of a composition for manipu-lation. GeNotator achieves this by allowing disjoint sections of a chromosome (known as *schema*) to be "frozen", and thus become dominant from generation to generation; only those genes that are not dominant are effected by the evolutionary process. For example, if a user wishes to only evolve a particular phrase, the hierarchic menu system can be used to track down which genes contribute to the phrase, and then force all other genes that do not contribute to it to become dominant.

6 6 GeNotator and the Internet

A current development has been to separate the Form Space Manager into a stand-alone ActiveX control for integration into html pages. Once embedded in an html page the control can be automated to upload published GSDs from any supplied URL address across the Internet.

The ActiveX control contains the familiar interactive "gardening" interface found in GeNotator and has enough functionality to compile and play phenotype versions of the composition assuming there are compatible MIDI tone generators attached to the hosting PC. Thus we now have an environment whereby a composer can publish a music genotype (GSD) on the Internet, and by doing so give it a life of it's own beyond the original composition.

An interesting consequence of this is that it provides a way to "democratize" the compositional process. Third parties, with little or no music knowledge, can be given the opportunity to develop other compositional ideas further and solely on the basis of personal taste. Genotype compositions that are published in this way become perpetual "works in progress", and the originating composer is able to discover with delight or horror the mutated offspring generative from their original conceptions.

7 Conclusions and Future Work

In so far that GeNotator provides a comprehensive approach to musical evolution within a hybrid compositional environment, there is much yet to explore. In terms of musical output, GeNotator can certainly evolve interesting music, but it is often a matter of balancing good starting conditions with a degree of form space bounding. The use of generative music grammars goes some way towards providing a mechanism with which to do this.

With the new enhancements to the music grammar and GUI, GeNotator has reached a point of functional maturity that makes composing with it relatively straight- forward, and most importantly, enjoyable. It is hoped the ActiveX control will also help the system reach a wider audience via the Internet.

Interesting applications beyond being simply a hybrid algorithmic environment include the prospect of royalty free generative music, or even the licensing of music genotypes in a particular musical style - something the advertising, web- content, and games/entertainment industries might find appealing.

Future developments will focus on automated grammar construction. Currently, the meta- composer still needs a degree of analytical aptitude in order to capture an interesting genotype structure definition and define a grammar, and it is perhaps here that initial enhancements should be focused. One avenue of interest is to provide automated grammar generation from existing music, or to apply the GA to the evolution of grammars themselves. GeNotator can currently be described as an "exploitation" evolutionary system: given an explicit grammar, it helps the user find interesting phenotypes within the constraints defined by that grammar. The next step is to provide for "exploration" evolution, i.e. evolving the grammar itself. Tentative steps are already being taken in this direction and it is hoped that the emerging field of genetic programming will provide further clues as to how this can be achieved in practice.

REFERENCES

[Biles, 1994] John A. Biles, *GenJam: A Genetic Algorithm for Generating Jazz Solos*. Proceedings of the 1994 ICMC.

[Chomsky, 1957] Noam Chomsky, *Syntactic Structures*, 1957.

[Dawkins, 1986] Richard Dawkins, *The Blind Watchmaker*. Penguin Books, 1986.

[Goldberg, 1989] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[Holtzman, 1981] S.R. Holtzman, *Using Generative Grammars for Music Composition*. Computer Music Journal 5, no. 1 (1981) pp. 51 - 64.

[Horner, 1995] Andrew Horner, *Wavetable Matching Synthesis of Dynamic Instruments with Genetic Algorithms*. Journal of the Audio Engineering Society, Vol. 43, No. 11, Novemberr 1995.

[Koza, 1992] J. R. Koza, *Genetic Programming - On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Latham *et al.*, 1992] William Latham, Stephen Todd. *Evolutionary Art and Computers*. Academic Press, 1992.

[Michalewicz, 1996] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Third edtion, Springer- Verlag, 1996.

[Plotkin, 1995] Henry Plotkin. *Darwin Machines and the Nature of Knowledge*. Penguin Books, 1995.

[Thywissen, 1992] Kurt Thywissen. *Evolving Melodies with Mu-Tonator*. M.Sc. programming assignment, Univ. of York, 1992.

[Thywissen, 1993] Kurt Thywissen. *GeNotator: An Environment for Investigating the Application of Genetic Algorithms in Computer Assisted Composition.* Univ. of York M.Sc. thesis, 1993.

[Thywissen, 1997] Kurt A. Thywissen , *Evolutionay Based Algorithmic Composition: A Demonstration of Recent Developments in GeNotator.* Proceedings of the 1997 ICMC.